Implementation and Performance Evaluation of Embedded IPsec in Microkernel OS

Mohammad Hamad

Institute of Computer and Communication Networks

Technical University Braunschweig
Braunschweig, Germany
mhamad@ida.ing.tu-bs.de

Vassilis Prevelakis

Institute of Computer and Communication Networks

Technical University Braunschweig
Braunschweig, Germany
prevelakis@ida.ing.tu-bs.de

Abstract—**The rapid development of the embedded systems and the wide use of them in many sensitive fields require safeguarding their communications. Internet Protocol Security (IPsec) is widely used to solve network security problems by providing confidentiality and integrity for the communications in the network, but it introduces communication overhead. This overhead becomes a critical factor with embedded systems because of their low computing power and limited resources. In this research, we studied the overhead of using embedded IPsec in constrained resource systems, which run microkernel operating system (OS), in terms of the network latency and throughput. To conduct our experiment first, we ran the test with an unmodified network stack, and then we ran the same test with the modified network stack which contains the IPsec implementation. Later, we compared the results obtained from these two sets of experiments to examine the overhead. Our research demonstrated that the overhead imposed by IPsec protocols is small and well within the capabilities of even low cost microcontrollers such as the one used in the Raspberry Pi computer.**

Keywords—Embedded IPsec; LwIP; network security; Microkernel.

I. Motivation

All the electronic devices around us such as cellphones, washing machines, cars, and aircraft rely on smaller systems called embedded systems. The use of embedded system produces many security issues. We will discuss some of these security issues later in this paper. These issues led us to focus on the security threats related to embedded systems. The use of microkernel OS can improve security in embedded systems [11]. The microkernel OS has many advantages such as: small kernel size which reduces the code that must be privileged, different access control privileges for each applications, and the ability to apply system policy to control applications' communications. The Genode framework [18] is a good example of microkernel OS which can run on embedded systems.

However, using a secure microkernel OS may solve some of the security problems of distributed systems, but it will not eliminate all of them. Most embedded systems rely on a network to communicate with each other. The use of wired and wireless connections in the embedded systems opens the possibility of many security vulnerabilities, for example, the security issues in internal network in aircraft. This network system contains three different functional networks: crew, passenger, and plane control networks. These networks separated by virtual LAN [24]. Many research have proved that the internal network is prone to security attack; passenger could use the on-board entertainment network to hack the control system of the aircraft [4].

The same issue appears in the internal network of vehicles. Most cars use the Controller Area Network (CAN) bus in their network communications for in-car network. Experimental security analysis of modern cars [21] showed that CAN protocol is vulnerable to denial of service attack and many more security issues. Since CAN packets are broadcasted to all nodes, any malicious component can observe, change, and send packets to any node [21], [3]. Some security experts used a security tool (CARSHARK) connected to CAN bus via OBD-II port to snoop on the car's communications. This attack was able to cause losing the control of the car by disabling the brakes or even by stopping the engine [21].

The security issues are not only related to wired communications; wireless communications also have inherent problems which may be used by the attackers to damage the systems. For example, there is a wireless pressure sensor in a car's tires, which measures the pressure and sends the measurement to the pressure central unit. An alert is sent to the car's central unit in case of low pressure. A hacker could, in some circumstances, interfere with the communications by sending false low-pressure reading or even stop the relevant ECU completely [12].

Another concern for the car security is to protect the driver and the vehicle information privacy and prevent unauthorized third parties from tracing and collecting the

This research has been sponsored by the Deutsche Forschungsgemeinschaft, under project "Controlling Concurrent Change."

DOI : 04.IJIS.2015.1.9

relevant information depends on vulnerabilities posed by wireless technologies which were used in the car [23]. In the previous example, an adjacent vehicle in the range of the wireless network able to receive the communication between the tire pressure sensor and the car's control unit, these communications may contain the ID of the sensor which can be used to trace the car and its location. Encrypting the messages is one solution to provide the required privacy [1] and reduce the ability to collect the car's identity information.

The previous examples indicate the importance of providing privacy, integrity, and origin data in the system communications. One solution to provide all these requirements is using IPsec. IPsec provides the integrity and the confidentiality for each packet in the communication session; moreover, IPsec runs in the third layer, which makes it transparent to upper layers, therefor applications do not need to have any knowledge of IPsec to be able to use it. IPsec based on two encapsulation protocols; one is Authentication Header (AH) which provides data-origin authentication, data-integrity, and protection from the replay-attack. The other one is Encapsulating Security Payload (ESP) which provides data confidentiality and also supports its own authentication scheme like that used in AH. IPsec can be used in either transport or tunnel mode. Transport mode is used to protect the data flows between a pair of hosts, while tunnel mode is used to protect the data flows between a pair of hosts, two gateways, or between a host and a getaway. IPsec provides security but in the same time, it affects the performance of communication because of the additional processing required. In embedded applications in particular, the main concern is that IPsec may cause unacceptable overhead.

In this paper, we measured the overhead and the performance of using IPsec in environment which contains embedded devices; these embedded devices run microkernel OSs. We compared our results with the case of running the test case without IPsec. We evaluated the measurements in the term of CPU utilization and communication delay. Finally, we examined the feasibility of applying IPsec in low price and limited resources embedded devices, which can be used in the future for in-vehicle network architecture.

The rest of the paper organized as follow. Section II describes the related work in measuring the performance of IPsec. Section III briefly details the prototype of the system. Section IV explains our implementation of the embedded IPsec. Section V discusses the test bed configurations. Section VI discusses the experimental results. Finally, Section VII contains some concluding remarks.

II. RELATED WORK

IPsec has been the subject of a lot of research with the object of building fast and secure systems. Many groups studied the performance of IPsec regarding different factors such as the network topology, the network protocols, and the OS of the platform in the test beds. Many researchers have studied the performance of IPsec under different OSs. Niedermayer et al. [16] studied the performance of IPsec under various kernel versions of Linux OS, they also measured the latency and the throughput when different encryption algorithms were used for ESP and different authentication algorithms were used for AH. They indicated that the use of the fast encryption algorithms will improve the performance of ESP. Miltchev et al. [31] studied the performance of IPsec under OpenBSD OS, they explored how various modes and various encryption algorithms affect its performance, they indicated that small packet sizes causes bad throughput. Narayan et al. [9] studied the performance of IPsec on Windows OS combinations, they measured the throughput of different versions of Windows OS (vista, XP, Windows7) when different IPsec algorithms were used (AES, DES, 3DES), their research showed that the performance of IPsec under different versions of Windows OS was comparable. In monolithic OSs the IPsec implementation is part of the kernel, it is interlacing with the other components of the kernel. The existence of IPsec in the kernel makes the trusted code base huge, it makes the IPsec also vulnerable to compromise if other components of the kernel were successfully penetrated. Härtig et al. [20] provided an approach to reduce the complexity of the trusted computing base of a VPN gateway while running it in the top of fiasco microkernel OS; they extracted the security relevant functions of IPsec and executed them in separate protection domain and let the untrusted components, which include the TCP/IP implementation, to interact with it. However, they did not provide any performance measurement on their implementation.

IPsec was also used to protect the communication in the networks which contain resource-constrained devices like sensors. Granjal et al. [25] studied the feasibility of using IPsec in context of Wireless Sensor Networks (WSN). In their study, they analyzed the execution times and memory requirements of cryptographic algorithms; they indicated that AES is the best choice for an encryption algorithm to be used within IPsec for the embedded devices. Raza et al. [19] evaluated the performance of IPsec implementation in WSN between the constrained resources sensors and a host computer, they studied the performance in terms of packet overhead and communication performance, they indicated that AH and ESP had similar response times for small packets. However, the response time for ESP increases as the packet size increases, they also indicated that using AES will improve processing time and reduce energy consumption comparing to the other encryption algorithms.

Comparing the IPsec performance overhead over IPv4 and IPv6 networks was the subject of [8]. They studied the performance of IPsec in terms of end to end throughput and delay parameters. The authors used Netperf as measurement tools and ran their experiment on ordinary PC, which was running OpenBSD as OS with 128 MB memory. They compared the result of using AH only, ESP only, AH and ESP, and without IPsec. The research indicated that the end to end throughput degrades to about 1/2 with the AH, while it degrades to about 1/4 with the ESP.

The overhead of IPsec on real-time interactive communications was presented in [5] and [10]. Both of the papers studied the influence of IPsec when it was used to protect voice or video transaction using a wireless link. They used IPsec in the tunnel mode with ESP. Zarrella et al.[2] showed how the added headers to each packet while

DOI : 04.IJIS.2015.1.9

applying IPsec cause an overhead on the communications. This overhead become significant especially for small VOIP packet; it can cause a considerable reduction in throughput over a low bandwidth wireless network. However, the results from [5] and [10] indicated that depending on an infrastructure, which has adequate bandwidth, the IPsec can be used to provide the security for VoIP with negligible difference in the quality of transmitted voice and video and with minimal decrease in the network performance.

In general, a lot of research studied the performance and the overhead of IPsec on powerful PCs running monolithic operating systems, where the IPsec is part of the kernel. On the other hand, fewer efforts have been done in studying the performance of IPsec when it was used in microkernel operating system, or when IPsec was used to protect the communication of resource constrained embedded devices. Thus, we established our test case by selecting embedded IPsec package and integrating it with lightweight IP stack, and then we used this modified package as a network stack in microkernel operating systems. We measured the overhead of using IPsec in this environment comparing to using native IP stack. The components of our system are discussed in details in Section III.

III. PROTOTYPE SYSTEM

This research is one effort in big project called Controlling Concurrent Change (CCC) [17]. The main goal of CCC project is to provide suitable methods and platform architectures for future car. This platform will enable the integration of updates and new functions in the field ensuring the quality, high robustness, security, and with no significant increase in cost or energy consumption. Security is one of the cornerstones to achieve the project's goals by providing the security for the upgrading and integration process. The first consideration is providing security for the communications of the architectural components through the local, closed inner-network or through open networks.

We simulated the ECUs inside the car by using very low cost platforms and then provide the security for their communications. We used Raspberry Pi [13] as a simulator of vehicle's ECU. Raspberry Pi has many attractive properties such as: it is a small size computer board composed of an ARM-like SoC, it includes all the I/O and storage interfaces, slot for a SD-card, Ethernet port and HMDI output. On the other hand, it has the ability to run microkernel OS like Genode. Moreover, we can get it at a low price approximately for tens of euros.

We used Genode framework as a runtime environment for many reasons; Genode supports various micro kernels such as Fiasco and Mini Linux, also it can run directly on the ARM hardware depend on a kernel called base-hw kernel, which provides the mechanism required by Genode. Genode consists of three layers: (1) the microkernel in the kernel space, (2) the Genode component in the user space, these components contains the devices drivers, resource multiplexer and protocol stacks and (3) at the top is the applications layer [22]. Genode is a secure operating system architecture, it maintains tree of processes and ensure that process is exposed to only those parts of system on which it ultimately depends, if one part of the system is compromised, the defect is limited to that particular part and its dependent parts, unrelated processes remain unaffected.

Genode rely on lightweight Internet Protocol (lwIP) [32] as TCP/IP stack. It ported lwIP in the form of a library running in user space. Basically, lwIP implementation focused on reducing the usage of the memory resources and the code size to make it suitable for the embedded systems. LwIP is a full-scale TCP/IP stack but without any implementation for the IPsec protocols. To gain our goal of providing secure communications we used embedded IPsec package and integrated it the lwIP stack. The implementation of this package is described in the next section.

IV. IPSEC IMPLEMENTATION

IPsec may be implemented by several different methods in a host. It can be implemented by “bump-in-the-stack” (BITS) method which means that IPsec is integrated directly below the IP layer. Another method of implementation is by integrating IPsec into the IP layer [6].

In our research, we used an existing package developed by Scherer and et al. [7]. They adopted the BITS method for their implementation by creating the IPsec as shim and inserted it between the IP and link layer as shown in Fig. 1.

This package has a number of deficiencies; it supports the tunnel mode only while the transport mode was not implemented. Moreover, it supports only the manual keying to set up the Security Association (SA) parameters. Finally, it does not handle the problem of IPsec with fragmentation; the package can handle the packet with size less than MTU. The previous implementation drops the packet silently In case the protected packet exceeds the MTU, this action may stop the whole communication later.

We enhanced their package by implementing the transport mode for both AH and ESP. We also removed the implementation of MD5, SHA1, and 3DES-CBC algorithms from the package since they are part of OpenSSL [27] library which is ported to the Genode framework. Adding fragmentation to the IPsec implementation could be achieved in two ways: (a) keep using the BITS architecture and re-implementing the fragmentation services to keep the IPsec package independent or (b) by integrating the IPsec into the IP layer and using the existing fragmentation code. We selected the second option in our implementation, as shown in Fig. 2.

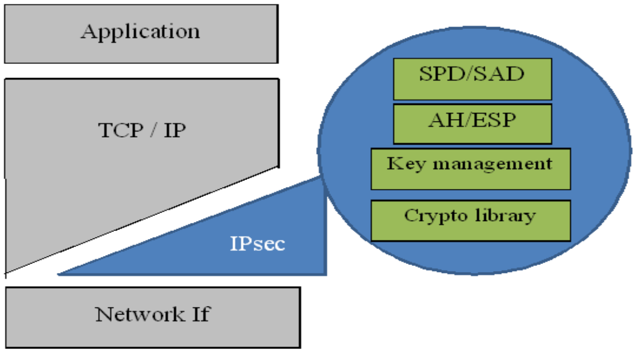

Fig. 1. Previous IPSec Implementation as layer between the IP and the network interface.

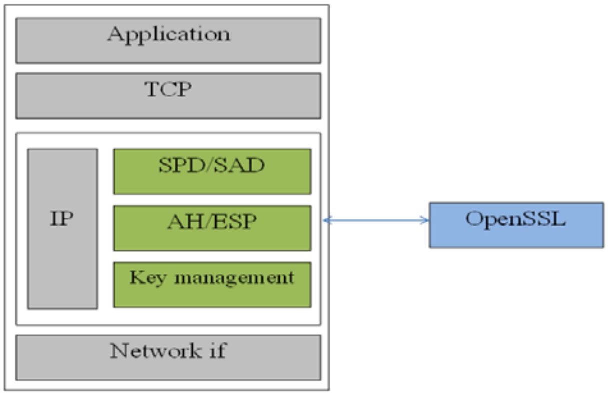

Fig. 2. Our IPSec Implementation, we integrate the IPsec within the IP layer.

We handled the fragmentation issue for outgoing packets as next:

a. We applied IPsec outgoing packets processing for all packets regardless of their sizes without checking the need of fragmentation.

a. We looked up for the SA depending on the source and destination addresses in the packet's header.

b. We processed the packet corresponding to the security protocol (AH, ESP, or both) which exists in the SA.

c. In the case of successful processing, we checked whether the packet needs to be fragmented to pass it to the fragmentation service which passes it later to the data link layer.

d. The packets, which did not need to be fragmented, were passed directly to the data link layer.

While for incoming packet we reassembled the fragmented packet before applying the IPsec processing on it:

a. We checked whether the incoming packet needs to be re-assembled or not. If the packet was fragmented then we passed it to the re-assembling service.

b. Then, we looked up for the SA based on the Security Index parameter (SPI) in the packet's header.

c. We processed the reassembled packet regarding to the security protocol (AH, ESP, or both) in the SA.

Our primary concern was the evaluation of the overhead of IPsec only without considering the overhead of Internet Key Exchange (IKE). Therefore, we did not implement the IKE in our system. Moreover, the manual keying is supported by IPsec implementation in the other system (i.e. OpenBSD) of our test bed, so we kept using manual keying to configure the cryptographic keys.

Interoperability is an important key in each software implementation. So, we tried to check our IPsec package's interoperability by testing it in non-homogenous environment. We were able to successfully establish connections to OpenBSD IPsec implementation.

V. Test bed Configuration

Measurement tools:

Many tools have emerged to aid in the performance monitoring of the networks. The most common tools to measure the latency are ping and Netperf [15]. Since ping works in the IP layer, its measurements do not consider the above layers. On the other hand Netperf is running in the application layer so its measurements will include most of the network stack. In addition, Netperf was ported in the Genode platform which gives us additional two reasons to prefer it: (a) confidence in its integrity with the other components of Genode environment and (b) confidence in our measurement results by comparing it with others' measurements results.

Netperf works as client/server application. It consists of two applications; the first one is the Netserver, which runs in the local host. It waits for the connection from the remote host. The remote side runs the second application, which called Netperf. Netperf provides us measurements for unidirectional throughput and end-to-end latency.

Experiment environment:

To conduct our experiment we used Raspberry Pi and Asus X53U laptop; Asus laptop was running OpenBSD 5.5 as OS and had a 1.65 GHz AM-E450 CPU. Raspberry Pi and Asus laptop were connected as shown in Fig. 3.

First, we ran the measurement with unmodified implementation of lwIP under Genode OS on the Raspberry Pi. Then, we ran the same test with our modified lwIP, which contains our embedded IPsec implementation, the test environment on the Raspberry Pi is described in Fig. 4. Later in the analysis section we compared the results from the two measurements. The IPsec configurations for the test bed were:

- IPsec Authentication: we configured the test bed to use MD5 as authentication algorithm.
- IPsec Confidentiality: we set up the test bed to use both MD5 as authentication algorithm and 3DES-CBC as encryption algorithm.
- IPsec for both AH and ESP was configured to use transport mode.

These configurations were hardwired in the IPsec code in the Raspberry Pi side and were configured by changing the ipsec.conf file on the OpenBSD side.

In our performance test, we considered the next network performance metrics: latency and end-to-end throughput.

DOI : 04.IJIS.2015.1.9

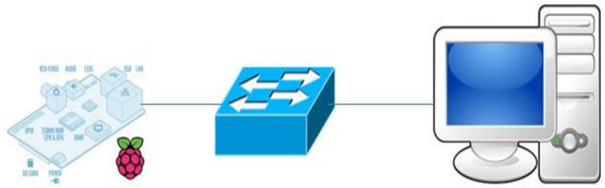

Fig. 3. Layout of experiment environment.

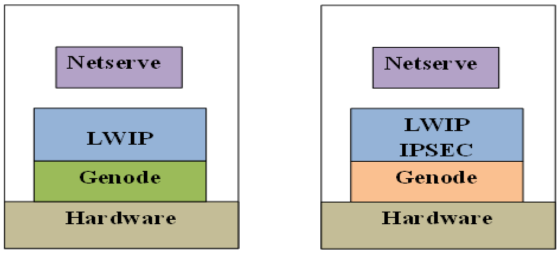

Fig. 4. Raspberry Pi running Genode without IPsec (left) and with modified lwIP stack which contains IPsec (right).

VI. Analysis of the Results

A. Latency

Latency is the time that a single packet takes to travel from the source to the destination. We measure the latency in two cases; (a) traffic flow from OpenBSD, which runs Netserver, to the Raspberry Pi, which runs Netperf, and (b) traffic flow from Raspberry Pi, which run Netserver, to OpenBSD, which run Netperf. Fig. 5 represents the relation between the latency and different packet sizes for the first case. We measured the latency by applying TCP_RR test from OpenBSD to Raspberry Pi with various message sizes while applying AH, ESP, and no-IPsec. The latency was calculated from the transaction rate that was provided from Netperf report. Fig. 6 shows the latency when we ran TCP_RR test in second case.

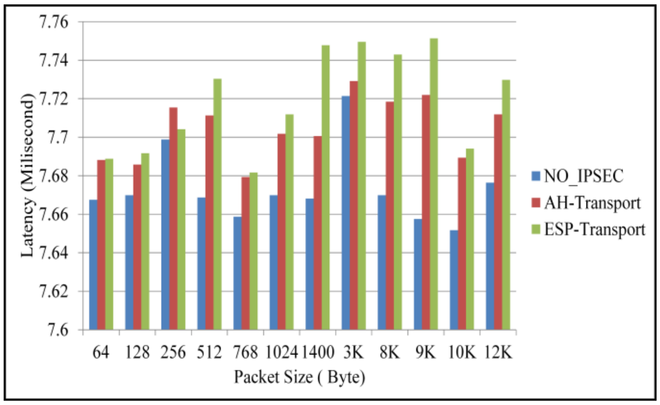

Fig. 5. Latency test for no-IPSec, AH transport, and ESP transport, traffics from OpenBSD to Raspberry Pi.

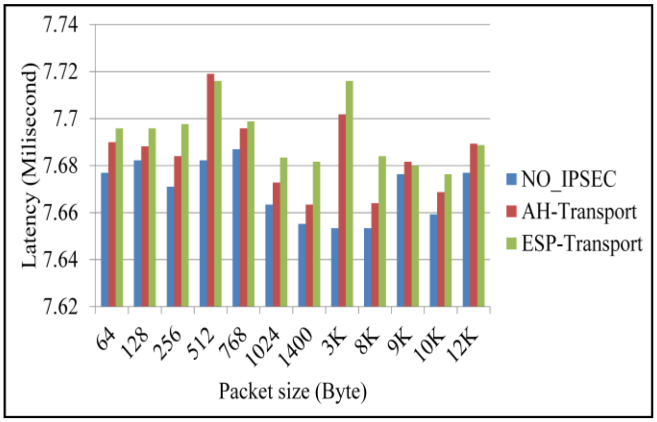

Fig. 6. Latency test for no-IPSec, AH transport, and ESP transport, traffics from Raspberry Pi to OpenBSD.

Both Fig. 5 and Fig. 6 indicate that latency, in the both direction, was increased when we apply ESP; but the difference in the latency values is still negligible. At the same time they indicate that latency is low when the flow runs from Raspberry Pi toward OpenBSD compared to the latency when the flow runs in the opposite direction.

B. Throughput

Throughput is the amount of data that can be transferred from the source to the destination in a specific period of time. Fig. 7 represents the result of testing throughput when the data flows from the server (Raspberry Pi) to the client (OpenBSD) in three different cases: IPsec with AH, IPsec with ESP, and no-IPsec. Different message sizes were used to measure the throughput for each test case.

Fig. 8 shows the throughput when the data flows from the client (OpenBSD) to the server (Raspberry Pi) in the same three cases. Both the Fig. 7 and Fig. 8 indicate that throughput decreases 33% when we use AH compared to its value without using IPsec, while it decreases 70% when we use ESP compared to the throughput value without using IPsec. They also indicate that throughput is better with the large message size. The low value of the throughput is not caused by our IPsec implementation; when we measure the throughput of the system without our IPsec code we got close result comparing to our result in the case of using no-IPsec with our implementation. To discover the reason of low throughput value we checked the packet loos issue, we found that the rate of packet retransmission was negligible (2%), so it cannot be the main reason of the low performance. However, the low throughput was expected with the current version of the USB driver and the way of processing the high interrupt load on the system. Optimizing the process of handling the interrupts issued by the Network Interface Card (NIC) will improve the performance of the network which consequently will reflect on the IPsec performance [28].

DOI : 04.IJIS.2015.1.9

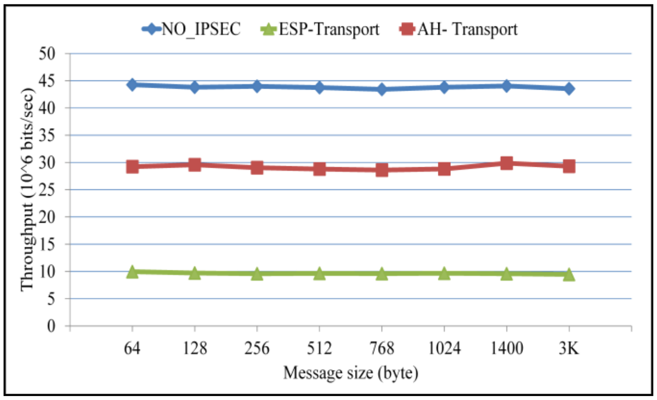

Fig. 7. Throughput test for NO IPsec, AH transport, and ESP transport. traffics from Raspberry Pi to OpenBSD.

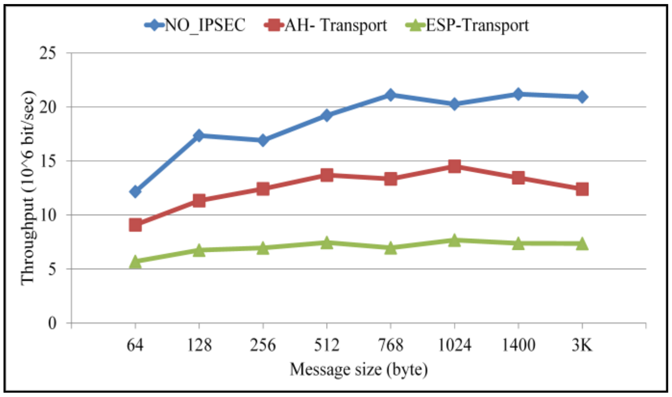

Fig. 8. Throughput test for NO IPsec, AH transport, and ESP transport. traffics from OpenBSD to Raspberry Pi.

VII. Conclusion

In this paper, we evaluated the performance of IPsec when it protects the communications of embedded systems. The embedded IPsec used as part of the network stack in microkernel framework running on the top of cheap and known hardware like Raspberry Pi.

Our experimental result indicated that the throughput value decreases when we use AH or ESP comparing to it is value in the case when no-IPsec was used. It indicates also that when the packet size is small, IPsec becomes less efficient. These results are comparable with the results of much other research [14] and [30]. Our results also showed that the increase of the latency value is small even after applying IPsec.

However, since most of the security issues in the distributed embedded system communications are related to the lack of proper authentication mechanism to prevent unauthorized parties from hijacking an embedded computer or sending false data [26]; we believe that applying AH in the communication of constrained resources platforms, such as Raspberry Pi running microkernel OS, will solve all the issues related to the missing of data origin authentication and the absent of data integrity with acceptable overhead on the efficacy. Achieving privacy by using IPsec with ESP in such constrained resource systems will cause significant overhead that might degrade the system capabilities in terms of throughput. We also believe that using more efficient encryption algorithm such as AES instead of 3DES algorithm will improve the performance of ESP [29].

Since Genode ported the lwIP stack in a way that we can use it as a user-level library for Genode applications, the lwIP stack will be linked against each networking application and they will share one single flow of control, as a future work we plan to study the performance of using only one lwIP stack with IPsec implementation and make all the network applications communicate with it via remote procedure calls (RPC). Another effort may be done by porting IKE to Genode framework to set up the security associations dynamically and then studying the additional overhead of using IKE in the same test bed.

References

[1] M. Wolf, A. Weimerskirch, and C. Paar, "Secure in-vehicle communication," in Embedded Security in Cars, New York: Springer Berlin Heidelberg, 2006, pp.95-109.

[2] H. Xiao and P. Zarrella. “Quality Effects of Wireless VoIP Using Security Solutions.” IEEE Military Commun. Conf. (MILCOM), 2004, vol. 3, pp. 1352-1357.

[3] M. Wolf, A. Weimerskirch, and C. Paar, "Security in Automotive Bus Systems," in Workshop on Embedded IT-Security in Cars, Bochum, Germany, 2004

[4] M.S. Ali, R. Bhagavathula, and R. Pendse, “Airplane Data Networks and Security Issues”, Digital Avionics Syst. Conf., Oct, 2004.

[5] J. Klaua and A. Hess, “ On impact of IPSec on Interactive Communication”, in Proc. 19th Int. Parallel & Distributed Process. Symp. (IPDPS), Denver, Colorado, 2005.

[6] S. Kent and K. Seo, “Security Architecture for the Internet Protocol”, RFC 4301, 2005.

[7] N. Schild and C. Scheuer, “Embedded IPSec, light weight IPSec Implementation”, Diplome Thesis, Berne Univ, Switzerland, 2003.

[8] S. Ariga, K. Nagahashi, M. Minami, H. Esaki, and J. Murai,”Performance evaluation of data transmission using IPSec over IPv6 networks,” in Proc. of the 10th Annu. Internet Society Conf. (INET 2000), Yokohama, Japan, July 2000.

[9] S. Narayan, M. Fitzgerald, and S. Ram, “Empirical Network Performance Evaluation of IPSec Algorithms on Windows Operating Systems Implemented on a Test-bed”, IEEE Int. Conf. on Computational Intell. and Computing Research (ICCIC), December 2010.

[10] M. Babu, “Performance analysis of IPSec VPN over VoIP network using OPNET,” Int. J. of Advanced Research in Comput. Sci. and Software, 2012, vol. 2, no. 9, pp. 38-42.

[11] G. Heiser, “Secure embedded systems need microkernel,” J. of System and Software, 2007.

[12] I. Roufa, R. Millerb, H. Mustafaa, T. Taylora, S. Ohb, W. Xua, M. Gruteserb, W. Trappeb, and I. Seskarb, ” Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study”, in Proc. Of 19th USENIX security Symp., Washington, DC, 2010, pp. 223-238.

[13] E. Upton and G. Halfacree, Raspberry Pi User Guide, 2nd ed. New York: Wiley, 2013.

[14] B.L. Chappell, D.T. Marlow, P. M. Irey IV , K. O'Donoghue, “An Approach for Measuring IP Security Performance in a Distributed Environment.“, in Proc. 11th IPPS/SPDP’99 Workshops , San Juan, Puerto Rico, USA,1999, pp. 389-394.

[15] Hewlett-Packard Company. Netperf: A network performance benchmark. http://www.netperf.org.

[16] H. Niedermayer, A. Klenk, and G. Carle, “The Networking Perspective of Security Performance - a Measurement Study,” in Proc. 13th GI/ITG Conf. Measurement, Modeling, and Evaluation of

DOI : 04.IJIS.2015.1.9

Comput. and Commun. Syst. (MMB), Nurnberg,Germany, 2006, pp. 119-136.

[17] Controlling Concurrent Change (CCC) Research unit, http://neu.ccc-project.org.

[18] Genode OS Framework, http://genode.org.

[19] S. Raza, S. D., A. Chung, D. Yazar, T. Voigt, and U. Roedig, “Securing communication in 6lowpan with compressed ipsec.” in Proc. 7th Int. Conf. on Distributed Computing in Sensor Syst. (DCOSS'11), Barcelona, Spain, 2011. pp.1 -8.

[20] H. Härtig, M. Hohmuth, N. Feske, C. Helmuth, A. Lackorzynski, F. Mehnert, and M. Peter, “The Nizza Secure-System Architecture.” In Proc. 1st Conf. on Collaborative Computing, Dec. 2005.

[21] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," in Proc. of the 31st IEEE Symp. on Security and Privacy, 2010, pp. 447-462.

[22] B. Pruthiviraj, G. S. Madhusuthun, S. Vijayasarathy, and K. Chakrapani, “A Microkernel Based Secure Operating system Using Genode Framework”, J. of Theoretical and Appl. Inform. Technology, Vol. 38, No. 2, 2012.

[23] U.S. Sen. Edward Markey (D-Mass.),” Tracking & Hacking: Security & Privacy Gaps Put American Drivers at Risk”, February 2015.

[24] N. R. Suresh and S. V. Mathew, “Security concerns for cloud computing in aircraft data networks.” in Proc. 6th Int. Conf. on Internet Technology and Secured Trans., (ICITST), Abu Dhabi, United Arab Emirates, 2011, pp. 132–136.

[25] J. Granjal, J. Sa Silva, E. Monteiro, J. Sa Silva, and F. Boavida, “Why is IPSec a viable option for wireless sensor networks,” in 5th IEEE Int. Conf. on Mobile Ad Hoc and Sensor Syst. (MASS 2008), Atlanta, GA, USA, 2008, pp. 802– 807.

[26] U. E. Larson and D. K. Nilsson "Securing vehicles against cyber attacks", Proc. 4th Annu. Workshop Cyber Security Inform. Intell. Research, CSIIRW\'08, vol. 288, 2008.

[27] The OpenSSL Project, "OpenSSL: The open source toolkit for SSL/TLS," April 2003, www.openssl.org.

[28] M. G. Iatrou, A. G. Voyiatzis, and D. N. Serpanos,” Network Stack Optimization for Improved IPsec Performance on Linux,” in Proc. Inte. Conf. on Security and Cryptography (SECRYPT 2009), Milan, Italy, 2009.

[29] M. Sokol, S. Gajewski, M. Gajewska, and L. Staszkiewicz, "Security and Performance Analysis of IPsec-based VPNs in RSMAD," in The 1st Int. Conf. on Advanced Commun. and Computation (INFOCOMP'11), 2011, pp. 70-74.

[30] J. Rossebø, J. Ronan, S.Davy, “An analysis of IPsec deployment performance in high and low power devices.”, 17th Nordic Teletraffic Seminar, Norway, August 2004.

[31] S. Miltchev, S. Ioannidis, and A. D. Keromytis ,” A Study of the Relative Costs of Network Security Protocols”, in Proc. of the FREENIX Track: 2002 USENIX Annu. Tech. Conf., Monterey, California, 2002.

[32] Light Weight TCP/IP stack, http://lwip.wikia.com.

DOI : 04.IJIS.2015.1.9