PANA applicability in constrained environments

Mitsuru Kanda <mitsuru.kanda@toshiba.co.jp>
Yoshihiro Ohba <yoshihiro.ohba@toshiba.co.jp>
Subir Das <sdas@appcomsci.com>
Stephen Chasko <Stephen.Chasko@landisgyr.com>

February 21, 2012

Constrained devices often have a need for access to a network for their intended services. They have some of the same security requirements for network access authentication as other network devices but the devices and environments that they operate within are quite different. These resource limited devices and the associated network environment pose additional challenges for adopting existing network access authentication protocol.

In this position paper, we discuss how an existing IETF[1] security protocol PANA [1] can be tailored to meet the requirements for constrained environments and recommend the extensions that are useful to make this protocol more suitable. Finally, we present a code footprint analysis of our PANA and EAP [2] implementations with some implementation guidelines for relatively constrained devices with 250KB ROM and 50KB RAM (per Class 2 device definition in [3]), with the conclusion that additional IETF work is required to support highly constrained devices with 100KB ROM and 10KB RAM (per Class 1 device definition in [3]).

PANA for Network Access Authentication in Constrained Environments

PANA is a network access authentication protocol that runs between a client (known as PANA Client (PaC)) and a server (known as PANA Authentication Agent (PAA)) that resides in the network. It defines an EAP (Extensible Authentication Protocol) lower layer and uses UDP [4] as transport with various operational options. The PANA protocol consists of four phases; Authentication and authorization phase, Access phase, Re-Authentication phase, and Termination phase. In addition PANA provides relay functionality called PANA Relay Element (PRE) [5] for multi-hop environment like wireless mesh network.
In order for PANA to be used in a constrained environment, following protocol level

[1] http://www.ietf.org

considerations and behaviors are required. It is important to note that in our design a PaC operates on a constrained device while the PAA is not assumed to operate on a constrained device.

A. Reduction of number of message exchanges

Reducing the number of message exchanges and message retransmission are critical for processing constrained, battery powered, bandwidth constrained devices and devices operating in an intermittent network. This also reduces the message processing cost on the constrained devices. In PANA, this reduction can be achieved in the following ways;

i. PANA Session initiation and re-authentication

There are two types of PANA session initiation, PaC-initiated session and PAA-initiated session. In a constrained device, it is suitable to support the PaC-initiated session and to not support the PAA-initiated session. One reason to not support PAA-initiated session is that constrained devices often operate in a low power mode that is not currently activated (also called a sleeping device). A sleeping device would not receive an unsolicited PANA-Auth-Request (PAR) message from a PAA (PAA-initiated session). For the Re-Authentication phase, the PaC and PAA must re-negotiate to keep the PANA session before the PANA session lifetime expires. In this Re-Authentication phase, it is suitable to support PaC-initiated re-authentication and not the PAA initiated re-authentication. This is also due to the sleepy device not being able to receive a PAR from a PAA.

ii. Piggybacking EAP message

EAP messages can be carried in PANA-Auth-Request (PAR) and PANA-Auth-Answer (PAN) messages. The PAN messages acknowledge the receipt of a message. “Piggybacking EAP” is the combination of the PAR message with the PAN message to reduce the number of message exchanges needed. Both the PaC and PAA should include EAP-Payload AVP in each of PANA-Auth-Request and PANA-Auth-Answer messages. Figure 1 shows an example "Piggybacking EAP" sequence of the Authentication and authorization phase. This ‘Piggybacking EAP’ technique can also be applied during the Re-Authentication phase.

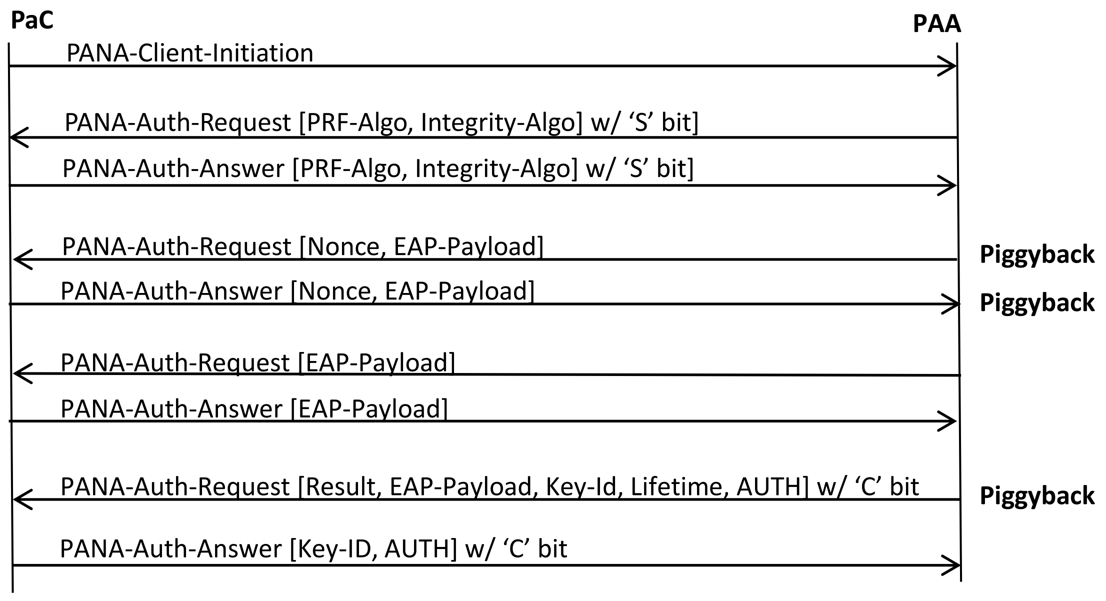
PaC
PAA
PANA-Client-Initiation
PANA-Auth-Request [PRF-Algo, Integrity-Algo] w/ ‘S’ bit]
PANA-Auth-Answer [PRF-Algo, Integrity-Algo] w/ ‘S’ bit]
PANA-Auth-Request [Nonce, EAP-Payload]
Piggyback
PANA-Auth-Answer [Nonce, EAP-Payload]
Piggyback
PANA-Auth-Request [EAP-Payload]
PANA-Auth-Answer [EAP-Payload]
PANA-Auth-Request [Result, EAP-Payload, Key-Id, Lifetime, AUTH] w/ ‘C’ bit
Piggyback
PANA-Auth-Answer [Key-ID, AUTH] w/ ‘C’ bit

Figure 1 Piggybacking EAP: An Example

iii. Handling ping message

During normal operation when a PANA session is established, both the PaC and PAA can send a PANA-Notification-Request (PNR) message with the ‘P’ (Ping) set for the peer’s PANA session liveliness check. This is also known as a “PANA ping”. On the other hand, in constrained environments, the PAA should not send a “PANA ping” message and should not rely on the PANA-Notification-Answer (PNA) with ‘P’ bit set message (“PANA ping” answer) from the PaC to check the PANA session liveliness. The reason for this restriction is that the PaC may be sleeping and would not send a “PANA ping” answer while sleeping in a low power mode. In order to check a PANA session liveliness, the protocol should allow other mechanisms for example; the PaC sends a “PANA ping” to the PAA every time the device wakes up.

iv. Session termination

The PaC and PAA should not send a PANA-Termination-Request (PTR) message except for explicitly terminating a PANA session within the session lifetime. Since both PaC and PAA know their own PANA session lifetime expiration, there is no need to send additional PANA-Termination-Request message when the PANA session lifetime is expired.

B. Choice of EAP methods

The EAP method which minimizes the number of messages is preferable. For example; EAP-GPSK [6] or EAP-PSK [7] exchanges seven messages while EAP-TLS [8] exchanges nine messages. EAP-GPSK or EAP-PSK are also preferred to EAP-TLS in environment where there are additional constrains such as, a lack of AAA[2] server connectivity or certificate usability.

C. Reduction of PDU[3] size

Here we recommend possible mechanisms to reduce PANA PDU size. The following two approaches are identified:

i. General Payload Compression:

IP payload compression (IPComp) [9] defines a mechanism to compress IP payload of any protocol. LZS-based compression [10] and LZJH-based compression [11] are available for IPComp. To utilize IPComp, two nodes first establish an IPComp Association (IPCA) between them [9]. The IPCA is established by dynamic negotiations or by manual configuration.

ii. Semantic Compaction for PANA:

Semantic compaction leverages the known data structure of a particular protocol (in this case, PANA). For example, unused fields (e.g., reserved fields, padding octets, etc.) can be removed from the compacted PANA payload. With semantic compaction more PDU size reduction is expected than general payload compression. An initial analysis on PANA message fields for which size reduction is possible is provided in Table 1 and Table 2.

Table 1 PANA Header size reduction

Fields that can be removed	Reserved
Fields that can be shortened	Message Type, Flags, Session Identifier, and Sequence Number

Table 2 PANA AVP size reduction

Fields that can be removed	Reserved, AVP Length (for fixed-length AVPs)
Fields that can be shortened	AVP Code, AVP Flags, AVP Length (for

[2] Authentication, Authorization and Accounting
[3] Protocol Data Unit

	variable-length AVPs), Vendor-Id, and AVP Value (for some AVPs of type OctetSting (with padding data), Integer32 or Enumerated)

We estimate that the PANA Header size can be reduced by 50% and PANA AVP size can be reduced by 18% to 80% except for EAP-Payload AVP for which the size can be reduced by less than 5% for 100-octet EAP message. (Detailed estimation is planned to be presented at the workshop.)

PANA implementation on constrained device

The minimization objective for this implementation mainly targets PaCs because constrained devices often are installed as network clients, such as sensors, metering devices, and control switches. The above techniques that reduce the number of messages and the size of messages are also effective for meeting the needs of the constrained devices. In this section, we describe additional mechanisms that will increase the implementation efficiency for a constrained environment..

A. Hash function

PANA uses AUTH_HMAC_SHA1_160 for PANA message integrity protection and PRF_HMAC_SHA1 for pseudo-random function (PRF). Both of these are based on SHA-1 hashing algorithm. Therefore we need to implement just one single hash function in the PaC. Since the SHA suite of functions is fairly common it can also be used for other functionality in the device. This allows us to operate in a constrained device with a smaller code footprint.

B. Sending PANA ping request

The PaC does not need to send a PANA ping request to the PAA periodically and may omit the PANA ping request feature itself if the PaC can detect the PANA session failure by other methods. For example, if there is a separate mechanism to detect network communication failure. This means the PaC does not need to implement the periodic liveliness check feature sending the PANA ping request. This optimization reduces the code size on the constrained device and reduces networking traffic on the constrained network.

C. EAP implementation

In general, there is no magic to reduce code size of EAP. An obvious optimization scheme is to reduce the number of supported EAP methods as much as possible. Although other optimization schemes are possible (e.g., sharing TLS session management among TLS-based EAP methods and other TLS-based applications), we believe that it is challenging to further reduce the code size of EAP without optimizing the design of EAP methods..

An implementation study

Here we explain the code size of our PANA implementation and study its applicability on constrained devices. The code size of our current prototype implementation is shown in Table 3. It should be noted that our current implementation is designed to be portable to various systems ranging from embedded operating systems to server class operating systems rather than targeted for a specific type of constrained devices. For this reason, our implementation contains extra code (e.g., #4, #5, #7, #8-1, #9, #10-1, #11-2 in Table 3) for abstraction and scalability related features such as, multiple PANA session handling. By removing the extra code it is possible to support relatively constrained devices with 250KB ROM and 50KB RAM as shown in Table 4. However, supporting constrained devices with 100KB ROM and 10KB RAM is still challenging. We recommend some approaches to overcome these challenges in the conclusion section.

Table 3 Sample implementation code size (Ubuntu 11.04 32bit, stripped)

#	Function			Code size
1	PANA message function			7.3KB
2	PANA AVP function			5.8KB
3	PANA I/O function			14.6KB
4	PANA session management function			15.0KB
5	PAA function			9.5KB
6	PaC function			9.7KB
7	PRE function			2.5KB
8	EAP Identify function	8-1	Authenticator	1.5KB
		8-2	Peer	1.3KB
		8-3	Total	2.8KB
9	EAP session management function			6.7KB

10	EAP-PSK function (cipher code is not included)	10-1	Authenticator	5.1KB
		10-2	Peer	5.1KB
		10-3	Total	10.2KB
11	EAP-TLS function (TLS code is not included)	11-1	Common	5.3KB
		11-2	Authenticator	1.0KB
		11-3	Peer	0.9KB
		11-4	Total	7.2KB

Note: The EAP-TLS function code size is smaller than the EAP-PSK function code size. But the actual EAP-TLS code size is much bigger than the ESP-PSK code size because additional code is necessary for TLS itself which can be hundreds of KB for OpenSSL[4] TLS implementation.

Table 4 Estimated PaC implementation code size

Implementation type	Estimated code size (total)
PaC (EAP-PSK) implementation (PAA/PRE/EAP-TLS code removed) (#1+#2+#3+#4+#6+#8-2+#9+#10-2)	65.5KB
Reduced PaC (EAP-PSK) implementation (PAA/PRE/EAP-TLS/multi-session code removed) (#1+#2+#3+#6+#8-2+#10-2)	43.8KB

Conclusion

In this position paper, we have shown that the PANA protocol can be implemented for relatively constrained devices with 250KB ROM and 50KB RAM by following implementation guidelines as described without affecting the existing PANA and EAP protocol behaviors.

However, we believe that additional work is needed in IETF to make PANA and EAP to be more suitable for application on constrained devices with 100KB ROM and 10KB RAM. We argue that by reducing PANA message size and PANA PDU compression alone cannot fulfill the constrained devices requirements; it also requires EAP method level compression. Additional design considerations on how to reduce EAP messages such as, omitting EAP-Identity exchange, how to support a single (existing or new) EAP

[4] http://www.openssl.org

authentication method with specific set of options and ciphersuites suitable for constrained environments are needed. Creating such profiles and document them in a guideline or best practice document may be helpful for the industry.

References

[1]. Ohba, Y. (Ed.), et al. "Protocol for Carrying Authentication for Network Access (PANA)", May 2008. http://www.ietf.org/rfc/rfc5191.txt. RFC5191

[2]. Levkowetz, H. (Ed.) et al. "Extensible Authentication Protocol (EAP), June 2004. http://www.ietf.org/rfc/rfc3748.txt. RFC3748

[3]. Bormann, C. "Guidance for Light-Weight Implementations of the Internet Protocol Suite", January 2012. http://www.ietf.org/id/draft-bormann-lwig-guidance-01.txt.

[4]. Postel, J. "User Datagram Protocol", August 1980. http://www.ietf.org/rfc/rfc768.txt RFC768

[5]. Duffy, P., et al. "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", August 2011. http://tools.ietf.org/rfc/rfc6345.txt.RFC6345

[6]. Clancy, T. and Tschofenig, H. "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", February 2009. http://tools.ietf.org/html/rfc5433. RFC5433

[7]. Bersani, F. and Tschofenig, H. "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", January 2007. http://tools.ietf.org/html/rfc4764. RFC4764

[8]. Simon, D., et al. "The EAP-TLS Authentication protocol", March 2008. http://www.ietf.org/rfc/rfc5216. RFC5216

[9]. Shacham, A. et al. "IP Payload Compression Protocol (IPComp)". September 2001, http://www.ietf.org/rfc/rfc3173.txt. RFC3173

[10]. Friend, R., et al. "IP Payload Compression Using LZS". December 1998, http://www.ietf.org/rfc/rfc2395.txt. RFC2395

[11]. Heath, J., et al. "IP Payload Compression Using ITU-T V.44 packet Method". January 2001, http://www.ietf.org/rfc/rfc3051.txt. RFC3051