Authentication components: Engineering experiences and guidelines

Pasi Eronen[1] and Jari Arkko[2]

[1] Nokia Research Center, pasi.eronen@nokia.com
[2] Ericsson Research NomadicLab, jari.arkko@nomadiclab.com

Abstract. Security protocols typically employ an authentication phase followed by a protected data exchange. In some cases, such TLS, these two phases are tightly integrated, while in other cases, such as EAP (Extensible Authentication Protocol) and Kerberos, they are separate and often implemented in different endpoints. However, careless application of this separation has lead to several vulnerabilities. In this paper we discuss reasons why this separation is often useful, what mistakes have been made, and what these mistakes have in common. We then describe some approaches how these problems could be avoided, especially focusing on EAP in wireless LANs. We also present some engineering observations that should be taken into account when designing reusable authentication components in the future.

1 Introduction

When designing a new system, a common engineering practice is to use existing components as building blocks. Typical examples of such components include not only things such as IP transport, TCP, XML and HTTP, but also components providing security services.

One well-known example is running HTTP over TLS. TLS uses certificates to authenticate the web server (usually client authentication is implemented by some other means), and provides encryption and integrity protection for the traffic. This case is rather simple, and its properties are easily understood even by those who are not intimately familiar with TLS. [3]

Other examples of reusable components include SASL and GSS-API [17, 18]. Both provide an extensible framework for authentication mechanisms, and may optionally provide also protection (encryption, integrity protection, replay protection, etc.) for application traffic. A common feature of all these approaches is that traffic protection (when present) is handled together with authentication and key exchange.

However, Kerberos separates authentication/key exchange from traffic protection [15], as does Extensible Authentication Protocol (EAP) which is used for Wireless LAN authentication, among other applications [7].

[3] Though occasionally some developers have forgotten that it is not enough to check that the server has a valid certificate—you must also check that the DNS name in the certificate matches the URL you are trying to access!

In this paper we explore issues related to this separation. We begin by exploring in Section 2 why this separation is often useful. In Section 3, we continue by presenting some examples of systems having this separation, and in Section 4 we analyze pitfalls found in some of them. In Section 5, we present how some of the pitfalls can be solved in EAP, and finally, Section 6 contains our conclusions from this work.

2 Reasons for separating authentication and traffic protection

It seems the main reason for separating authentication and traffic protection is that they can be distributed to separate endpoints. The following sections discuss various reasons why this may be desirable.

2.1 Centralized storage of secrets

When authentication is based on shared secrets (or some other non-public-key based credentials), it is convenient to store the secrets in a centralized place, instead of having copies of them around in all nodes that need to do authentication.

In Kerberos, this would be called a Key Distribution Center (KDC); in EAP, it is called AAA server (authentication, authorization and accounting); in cellular networks, it is Authentication Center (AuC).

In cellular networks, a somewhat similar separation is also present on the client side: authentication is done by a smartcard (Subscriber Identity Module or SIM), which hands the traffic encryption keys over to the mobile phone. However, this situation is quite simple since a single smartcard is always inserted in a single phone.

2.2 Centralized authorization

PKI could be used for decentralized authentication, without requiring an online authentication server. However, current PKIs are not that well suited to authorization; either each service has to handle its own authorization information, or a centralized server for authorization is needed. Often authentication and authorization information are combined in a single AAA server.

2.3 Centralized audit trail

When authentication uses a centralized KDC, this leaves an audit trail at the KDC. This makes e.g. detection of compromised credentials or other "fraud detection" easier.

This evidence of authentication is also useful for charging-related purposes. In typical roaming environments, the visited operator reports how much services the user has used, and is paid according to pre-agreed rates. Having evidence

of authentication in the home operator AAA server prevents a visited operator from fabricating charges for users who have never been on the network (of course, the visited operator may still report inflated usage figures).

2.4 Extensibility of authentication

Experience has shown that there is no single authentication method suitable for all environments. Centralized authentication server simplifies implementations since the nodes providing the various services do not need to implement all possible authentication methods.

For instance, IEEE 802.11i wireless LAN [11] access points typically implement only simple shared-secret authentication themselves, and everything else (such as passwords, token cards, certificates, or Kerberos) is delegated to an AAA server that actually implements the authentication protocol.

Another aspect of extensibility is the service-to-KDC authentication. While e.g. Kerberos supports public-key based authentication of users, services and KDC must still share a symmetric key. EAP, on the other hand, allows this authentication to be chosen independently of user authentication method.

For instance, in many environments a "transitive" authentication and authorization via AAA proxies is enough (and provides scalability when e.g. the number of WLAN access points around the world can be huge). This works especially in environments where e.g. a home (KDC) operator just needs to know that that the request came from network X, but not which from WLAN access point in X.

2.5 Different protocol layers

Performance issues can sometimes dictate that the endpoints for authentication and data protection are different, or at least that they are implemented on a different layer. For instance, in NFSv4, it would be desirable to delegate the traffic protection to the IPsec layer with widely available hardware acceleration, even if for flexibility and access control reasons the authentication has to reside at the application.

3 Case example

Figure 1 shows how EAP is used in typical wireless LAN environment. The EAP conversation itself (Step 1) is between the client and the AAA server. EAP payloads are transported using EAP-over-LAN from the laptop to the access point (AP), and over EAP-over-RADIUS or Diameter from the AP to the AAA server [2, 10]. The client and AAA server authenticate each other using some EAP method; for instance, EAP-TLS uses public-key certificates.

The AAA server then determines whether the user should be granted access or not, and if access is granted, sends a Diameter-EAP-Answer message (Step 2) to the AP, containing a successful result code, the client's identity, and

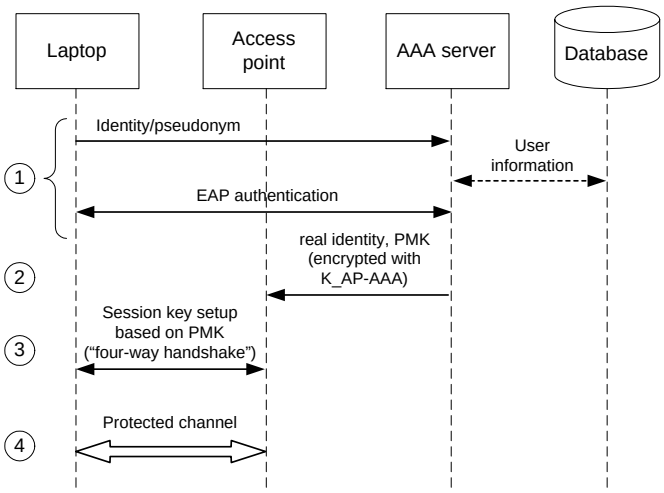

Fig. 1. EAP authentication in typical wireless LAN situation. AAA server and database are often combined in a single element.

the Pairwise Master Key (a session key that was established during the EAP conversation). There is also a second authentication taking place: the Diameter conversation between the AP and the AAA server is protected using, e.g., TLS. Thus, the PMK is encrypted using a key shared between the AP and AAA server.

After the AP has received the key and permission to grant access, it initiates the 802.11i "four-way handshake" (Step 3); basically a nonce exchange that also protects the ciphersuite negotiation that took place earlier. This creates fresh keys that are then used to encrypt the traffic and protect its integrity (Step 4).

Figure 2 shows similar scenario for UMTS. The protocols used are completely different (and the figure omits many details), but the same steps can be identified. Kerberos, shown in Figure 3 handles issues such as replay protection differently, but again, there are many similarities.

4 Analysis and problems

4.1 Service identities and lack of them

In Kerberos, the client requests a ticket for a particular server (identified by a principal identifier, usually comprising of service type and host name) from the KDC, and when the session key is used to protect communication, the client can be assured that it is talking to the intended server (assuming, of course, that the KDC is trustworthy and the server in question has not been compromised).

Currently EAP has no concept of "service identity", an identifier that would be authenticated to the client: the AAA server might have an identity, but not

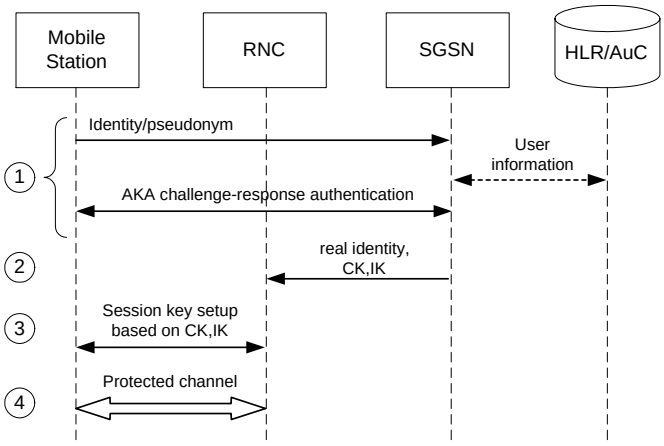

Fig. 2. UMTS authentication for packet switched access (simplified).

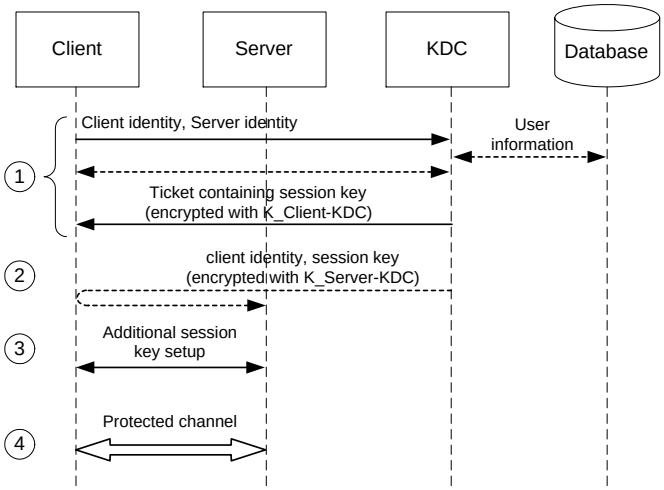

Fig. 3. Kerberos authentication (simplified). Note the Server identity sent by the client.

the node providing the service. That is, after a successful EAP authentication, the client knows it is talking to *some* network element trusted by the AAA server, but not which one [3]. The network element can, of course, tell the client its identity, but this information is not verified by the AAA server; or in other words, a malicious or compromised server could lie about its identity.

Cellular network authentication mechanisms such as UMTS AKA have a similar limitation. This "feature" probably has historical reasons behind it. For instance, in UMTS the cell is uniquely identified by a digit sequence called Cell Global Identification (CGI) [22]. However, since the mobile terminal does not really know which CGI it should be talking to (simplifying things a bit, it is usually the one with the best signal strength), CGI is not authenticated. (The CGI is not something that the user would enter or ever see, unlike in Kerberos where the host name is often selected by the user)

Probably it was also assumed that a compromise of a base station would be a rare event, and even if that happened, the intruder would not be interested in impersonating some other CGI towards the client (because doing so would not have any real advantage over continuing to use the real CGI value).

Somewhat similar thing can be found in EAP. In WLANs, the access points are uniquely identified by BSSID, a 48-bit random-looking string that is not meaningful to the user, since BSSID is chosen based on signal strength. Successful EAP/802.11i authentication ensures that the user is talking to some WLAN AP part of the e.g. corporate WLAN, but not which one—again, authenticating this identifier is not that important if the client cannot anyway tell the difference. Similarly, it might have been assumed that an attacker who compromises an access point would be interested in hacking into the corporate intranet rather than masquerading as some other AP towards the client.

The common feature here is that UMTS AKA and EAP consider authentication as having two different parties: the client and "the network". Internally, the network may have different nodes with different roles (e.g. UMTS authentication information is stored in AuC), but this structure is not directly visible to the client.

This situation is quite different from e.g. Kerberos, where service identities (such as host names) are usually very visible to the client, and different nodes are offering radically different types of service.

4.2 Attacks across protocols

While in many cases the client is not interested in the exact identifier of a node providing some particular service, it can at least know the type of service it is expecting. If this service type is not authenticated, the effects of security problems can "spread" from one service type to another. For instance, in EAP a compromised IKEv2 gateway can impersonate a WLAN access point, if same user and AAA server credentials are used for both.

A different example is the GSM A5/2 attack [6]. An attacker can use this vulnerability in GSM air interface encryption to masquerade as a WLAN client with EAP-SIM [12, 9]. Similarly, if it were later found that, for instance, IKEv2 has serious weaknesses[4], this could be used to attack WLANs using the same credentials.

[4] We do not believe this is likely, and this example is for illustrative purposes only.

There have been proposals to add a "context field" to EAP methods, communicating the type of service to be provided [21, 16]. Even without the concept of service identities, this would prevent attacks from one type of service to another. For some EAP methods, it may also be feasible to include context information in a certificate presented by the AAA server [13].

The "context field" is quite similar to "Special RANDs" that have been proposed for GSM. In this case, it is not possible to add a new field, so a couple of bits from the random challenge (RAND) field are used for that [20, 19].

4.3 Problems in protocol composition

Problems can also occur when several components providing various functions are composed together in a system. Asokan et al. [5] presented several examples where having one component responsible for server authentication and another one for client authentication could result in man-in-the-middle vulnerabilities.

5 Adding service identities to EAP

In addition to adding a "context field" to EAP, there have been some discussion about adding proper service identities as well.

While in some cases, authenticating the service as being part of the legitimate "network" is enough, there are cases where the client could be interested in parts of the network structure, and could, in some cases at least, tell the difference between nodes that it is "supposed to be" talking to and nodes that it is not interested in talking to. This is emphasized if some parts of the network are in fact less trustworthy than others.

For instance, in wireless LANs, the client could also know the Service Set Identifier (SSID) of the WLAN. This is a usually human-readable string identifying some set of access points, and in many cases the WLAN client software shows it to the user, or asks the user to select the right SSID from a list.

We are currently working on a mechanism to add this functionality (sometimes called "channel bindings" and "connection bindings") to existing EAP methods in an extensible way [4]. The basic idea is to embed a "context block" to an EAP messages that are protected by the method-specific integrity check. This context block would contain not only the service type (e.g. "IEEE 802.11i wireless LAN") but also service identities each party could be interested in. For wireless LANs these could be BSSID and SSID, and possibly also a human-readable network name (e.g. "Joe's Coffee Shop, Memphis, Tennessee").

Actually implementing this requires the ability to connect various types of identities. For instance, if Diameter protocol over TLS is used between the WLAN access point and AAA server, the identities that are authenticated are usually the DNS names of the parties. However, the WLAN client is unlikely to be interested in the DNS name of the WLAN access point, which could be something quite cryptic such as "ap1733.mph.tn.example.net". Therefore, the AAA server has to be able to map this securely to more meaningful identities such as

"Joe's Coffee Shop, Memphis, Tennessee". This more meaningful identifier could then be sent to the client in the "context block".

This approach extends what is possible in Kerberos: there it is usually assumed that the service identifier is a host name (i.e., a DNS name).

6 Conclusions

In this paper we have analyzed the reasons why separating authentication from traffic protection is often desirable, and how this has been implemented in various systems. Many of these implementations have gotten things wrong, often due to historical reasons. However, it may not be fair to say that people have been reinventing the wheel badly—often they have tried to solve problems that better-designed systems, such as Kerberos, did not solve.

One of these real-world requirements is the choice of authentication mechanisms. While from an abstract viewpoint, PKIs might be more "secure" (whatever that means) than passwords, in reality security is mostly about other things than cryptography—and then issues such as comparing required investments with reduction in risks are appropriate criteria.

One could also say that politics has had a lot to do with many of these problems: sometimes there has been reluctance to modify protocols to better answer real-world requirements, so to avoid spoiling "clean" protocols with messy real-world details.

Consider the case of Internet Key Exchange: Original IKE did not support authentication based on token cards or one-time passwords. Since then, there has been various efforts (XAUTH, CRACK, PIC, IKEv2), but after more than five years of discussion, there is still no standardized way to e.g. authenticate users with token cards (vendors certainly do have their own proprietary solutions). All along, the argument seems to have been "all password-based authentication is insecure; IPsec is designed to be secure; therefore, you have to deploy a PKI for it".

We do not agree with this. While it does not make sense to engineer bad security solutions, it does make sense to engineer security solutions that solve real-world problems, rather than require the real-world problem to adapt to some abstract notion of "perfect security". We believe that authentication mechanisms are an example of a "tussle space" [8]; something that should not be hardcoded into architectures or protocols, but instead left open to choice.

Related to the choice of authentication methods is also the choice of identifier types. We argued earlier that e.g. in wireless LANs, the DNS name of the WLAN access point is not a meaningful identifier for the WLAN client. Since ultimately these identities should be somehow connected to the *intent* of some person, different applications should be able to use different identities, and a single service may require multiple identities of different types.

These considerations suggest that we will see more systems separating authentication and traffic protection, some of them probably repeating the same mistakes already made. The examples we have used (EAP, wireless LANs, and

cellular networks) certainly did not get things right the first time. In this paper we have highlighted some similarities between these mistakes in hope that they could be avoided in the future.

Acknowledgments

We would like to thank N. Asokan and Valtteri Niemi for their valuable comments and suggestions.

References

1. Martin Abadi and Roger Needham, "Prudent Engineering Practice for Cryptographic Protocols", *IEEE Transactions on Software Engineering* 22(1):6–15, 1996.
2. Bernard Aboba and Pat Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, 2003.
3. Bernard Aboba, Dan Simon, Jari Arkko, and Henrik Levkowetz, "EAP Key Management Framework", work in progress (IETF Internet-Draft draft-ietf-eap-keying-01.txt), 2003.
4. Jari Arkko and Pasi Eronen, "Authenticated Service Identities for the Extensible Authentication Protocol (EAP)", work in progress, 2004.
5. N. Asokan, Valtteri Niemi, and Kaisa Nyberg: "Man-in-the-Middle in Tunnelled Authentication Protocols", *Proc. International Workshop on Security Protocols 2003*.
6. Elad Barkan, Eli Biham, and Nathan Keller, "Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication", *Proc. CRYPTO 2003*.
7. Larry Blunk, John Vollbrecht, Bernard Aboba, James Carlson, and Henrik Levkowetz, "Extensible Authentication Protocol (EAP)", work in progress (IETF Internet-Draft draft-ietf-eap-rfc2284bis-09.txt), 2004.
8. David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet", *Proc. ACM SIGCOMM 2002*.
9. Ericsson and TeliaSonera, "Implications of the A5/2 Attack for 3GPP WLAN Access", 3GPP TSG SA3 working document S3-030733, 2003.
10. Pasi Eronen, Tom Hiller, and Glen Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", work in progress (IETF Internet-Draft draft-ietf-aaa-eap-05.txt), 2004.
11. Institute of Electrical and Electronics Engineers, "IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements", work in progress (IEEE Draft P802.11i/10.0), 2004.
12. Henry Haverinen and Joseph Salowey, "EAP SIM Authentication", work in progress (IETF Internet-Draft draft-haverinen-pppext-eap-sim-13.txt), 2004.
13. Russ Housley and Tim Moore, "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)", RFC 3770, 2004.
14. Charlie Kaufman (ed.), "Internet Key Exchange (IKEv2) Protocol", work in progress (IETF Internet-Draft draft-ietf-ipsec-ikev2-13.txt), 2004.

15. John Kohl and Clifford Neuman, "The Kerberos Network Authentication service (V5)", RFC 1510, 1993.
16. Hugo Krawzcyk, "Changes to EAP methods", message on eap@frascone.com mailing list 2003-09-03, http://mail.frascone.com/pipermail/public/eap/2003-September/001638.html.
17. John Linn, "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, 2000.
18. John Myers, "Simple Authentication and Security Layer (SASL)", RFC 2222, 1997.
19. Nokia, "Using Special RANDs to separate WLAN and GSM/GPRS", 3GPP TSG SA3 working document S3-040100, 2004.
20. Orange and Vodafone, "Introducing the special RAND mechanism", 3GPP TSG SA3 working document S3-030698, 2003.
21. Jose Puthenkulam, Victor Lortz, Ashwin Palekar, and Dan Simon, "The Compound Authentication Binding Problem", work in progress (IETF Internet-Draft draft-puthenkulam-eap-binding-04.txt), 2003.
22. 3rd Generation Partnership Project, "Technical Specification Group Core Network; Numbering, addressing and identification (Release 5)", 3GPP Technical Specification 23.003 V5.8.0, 2003.
23. 3rd Generation Partnership Project, "Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (Release 5)", 3GPP Technical Specification 33.102 V5.1.0, 2002.