Internet Engineering Task Force (IETF)
Request for Comments: 7029
Category: Informational
ISSN: 2070-1721

S. Hartman
M. Wasserman
Painless Security
D. Zhang
Huawei

October 2013

Extensible Authentication Protocol (EAP) Mutual Cryptographic Binding

Abstract

As the Extensible Authentication Protocol (EAP) evolves, EAP peers rely increasingly on information received from the EAP server. EAP extensions such as channel binding or network posture information are often carried in tunnel methods; peers are likely to rely on this information. Cryptographic binding is a facility described in RFC 3748 that protects tunnel methods against man-in-the-middle attacks. However, cryptographic binding focuses on protecting the server rather than the peer. This memo explores attacks possible when the peer is not protected from man-in-the-middle attacks and recommends cryptographic binding based on an Extended Master Session Key, a new form of cryptographic binding that protects both peer and server along with other mitigations.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7029.

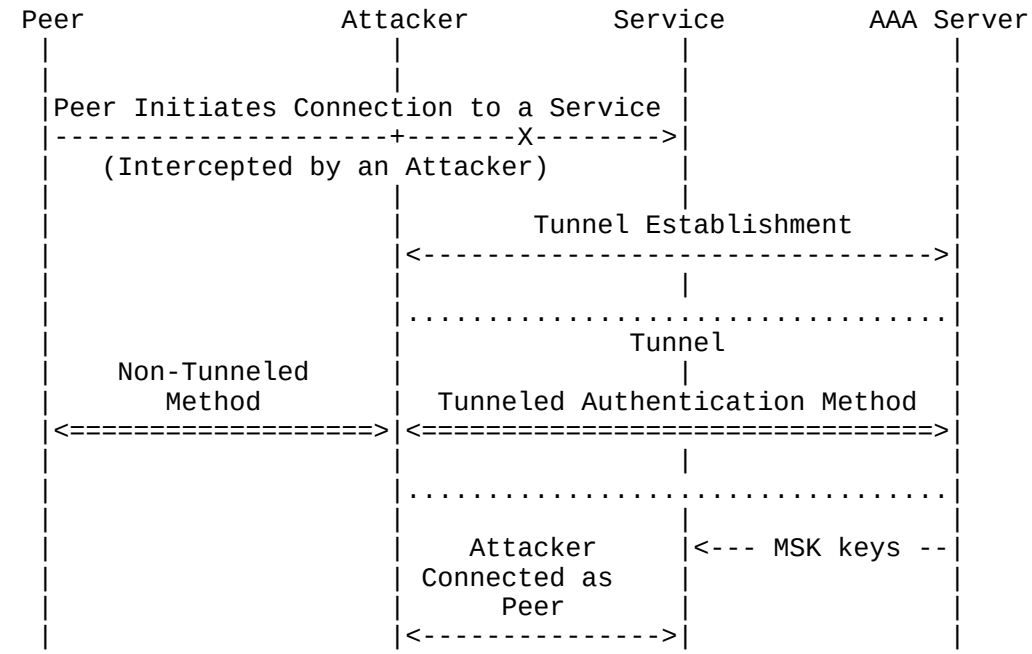
Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP) [RFC3748] provides authentication between a peer (a party accessing some service) and a authentication server. Traditionally, peers have not relied significantly on information received from EAP servers. However, facilities such as EAP channel binding [RFC6677] provide the peer with confirmation of information about the resource it is accessing. Other facilities such as EAP Posture Transport [PT-EAP] permit a peer and EAP server to discuss the security properties of accessed networks. Both of these facilities provide peers with information they need to rely on and provide attackers who are able to impersonate an EAP server to a peer with new opportunities for attack.

Instead of adding these new facilities to all EAP methods, work has focused on adding support to tunnel methods [RFC6678]. There are numerous tunnel methods, including [RFC4851] and [RFC5281], and work on building a Standards Track tunnel method [TEAP]. These tunnel methods are extensible. By adding an extension to support a facility such as channel binding to a tunnel method, an extension can be used with any inner method carried in the tunnel.

Tunnel methods need to be careful about man-in-the-middle attacks. See [RFC6678] (Sections 3.2 and 4.6.3) and [TUNNEL-MITM] for a detailed description of these attacks. For example, an attack can happen when a peer is willing to perform authentication inside and outside a tunnel. An attacker can impersonate the EAP server and offer the inner method to the peer. However, on the other side, the attacker acts as a man-in-the-middle and opens a tunnel to the real EAP server. Figure 1 illustrates this attack. At the end of the attack, the EAP server believes it is talking to the peer. At the inner method level, this is true. At the outer method level, however, the server is talking to the attacker.



A classic tunnel attack where the attacker inserts an extra tunnel between the attacker and EAP server.

Figure 1: Classic Tunnel Attack

There are two mitigation strategies for this classic attack. First, security policy can be set up so that the same method is not offered by a server both inside and outside a tunnel. Second, a technical solution is available if the inner method is sufficiently strong: cryptographic binding is a security property of a tunnel method under which the EAP server confirms that the inner and outer parties are the same. Cryptographic binding is typically implemented by requiring the outer party (the other end of the tunnel) to prove knowledge of the Master Session Key (MSK) of the inner method. This proves to the server that the inner and outer exchanges are with the same party. RFC 3748's definition of cryptographic binding allows for an optional proof to the peer that the inner and outer exchanges are with the same party. As discussed below, proving knowledge of the MSK is insufficient to prove to the peer that the inner and outer exchanges are with the same party.

1.1. Keywords for Requirement Levels

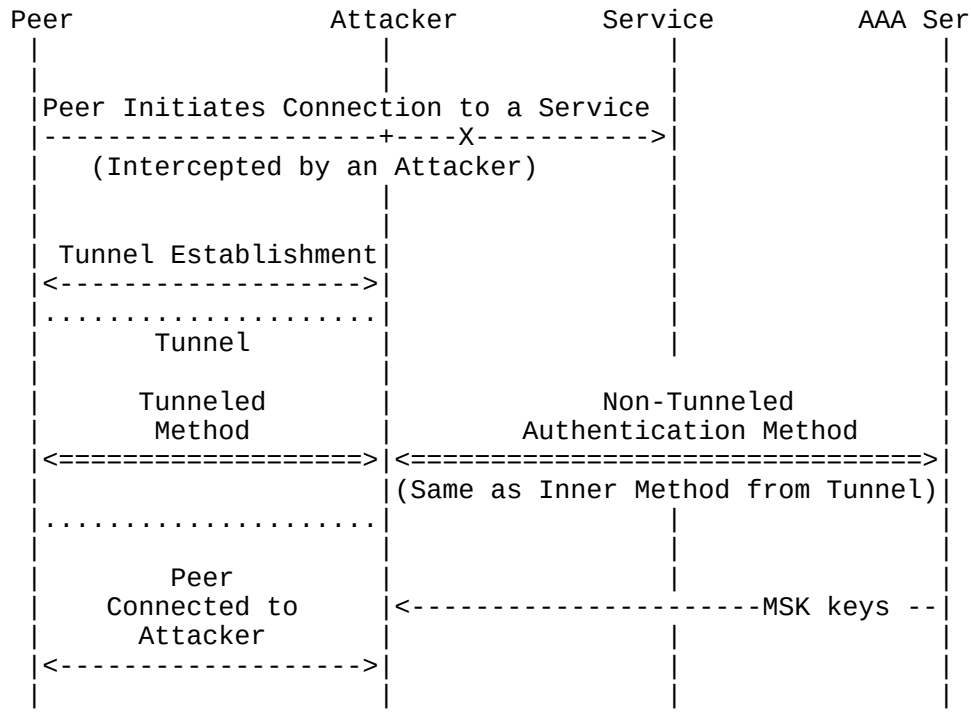
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. An Example Problem

The GSS-EAP (Generic Security Service Extensible Authentication Protocol) mechanism [GSS-EAP] provides application authentication using EAP. A peer could reasonably trust some applications significantly more than others. If the peer sends confidential information to some applications, an attacker may gain significant value from convincing the peer that the attacker is the trusted application. Channel bindings are used to provide information to the peer about the application service to which the peer connects. Prior to channel bindings, peers could not distinguish one Network Access Service (NAS) from another, so attacks where one NAS impersonated another were out of scope. However, channel bindings add this capability and thus expands the threat model of EAP. The GSS-EAP mechanism requires distinguishing one service from another.

Consider the following example. A relatively untrusted service, say a print server, has been compromised. A user is attempting to connect to a trusted service such as a financial application. Both the print server and the financial application use an Authentication, Authorization, and Accounting protocol (AAA) to transport EAP authentication back to the user’s EAP server. The print server mounts a man-in-the-middle attack on the user’s connection to the financial application and claims to be the application.

The print server offers a tunnel method towards the peer. The print server extracts the inner method from the tunnel and sends it on towards the AAA server. Channel binding happens at the tunnel method though. So, the print server is happy to confirm that it is the financial application. After the inner method completes, the EAP server sends the MSK to the print server over the AAA protocol. If only the MSK is needed for cryptographic binding, then the print server can successfully perform cryptographic binding and may be able to impersonate the financial application to the peer.



A modified tunnel attack when an extra server rather than extra client is inserted.

Figure 2: Channel Binding Requires More than Cryptographic Binding

This attack is not specific to GSS-EAP. The channel bindings specification [RFC6677] describes a number of situations where channel bindings are important for network access. In these situations, one NAS could impersonate another by using a similar attack.

3. The Server Insertion Attack

The previous section described an example of the server insertion attack. In this attack, one party adds a layer of tunneling such that from the perspective of the EAP peer, there are more methods than from the perspective of the EAP server. This attack is most beneficial when the party inserting the extra tunnel is a legitimate NAS, so mitigations need to be able to prevent a legitimate NAS from inappropriately adding a layer of tunneling. Some deployments utilize an intentional intermediary that adds an extra level of EAP tunneling between the peer and the EAP server; see Section 3.3 for a discussion.

3.1. Conditions for the Attack

For an inserted server attack to have value, the attacker needs to gain an advantage from its attack. An attacker could gain an advantage in the following ways:

- The attacker can send information to a peer that the peer would trust from the EAP server but not the attacker. Examples of this include channel-binding responses.
- The peer sends information to the attacker that was intended for the EAP server. For example, the inner user identity may disclose privacy-sensitive information. The channel-binding request may disclose what service the peer wishes to connect to.
- The attacker may influence session parameters. For example, if the attacker can influence the MSK, then the attacker may be able to read or influence session traffic and mount an attack on the confidentiality or integrity of the resulting session.
- An attacker may impact availability of the session. In practice though, an attacker that can mount a server insertion attack is likely to be able to impact availability in other ways.

For this attack to be possible, the following conditions need to hold:

1. The attacker needs to be able to establish a tunnel method with the peer over which the peer will authenticate.
2. The attacker needs to be able to respond to any inner authentication. For example, an attacker who is a legitimate NAS can forward the inner authentication over AAA towards the EAP server. Note that the inner authentication may not be EAP.
3. Typically, the attacker needs to be able to complete the tunnel method after inner authentication. This may not be necessary if the attacker is gaining advantage from information sent by the peer over the tunnel.
4. In some cases, the attacker may need to complete a Secure Association Protocol (SAP) or otherwise demonstrate knowledge of the MSK after the tunnel method successfully completes.

Attackers who are legitimate NASes are the primary focus of this memo. Previous work has provided mitigation against attackers who are not NASes; these mitigations are briefly discussed.

3.2. Mitigation Strategies

3.2.1. Server Authentication

If the peer confirms the identity of the party that the tunnel method is established with, the peer prevents the first condition (attacker establishing a tunnel method). Many tunnel methods rely on Transport Layer Security (TLS) [RFC5281] [TEAP]. The specifications for these methods tend to encourage or mandate certificate checking. If the TLS certificate is validated back to a trust anchor and the identity of the tunnel method server confirmed, then the first attack condition cannot be met.

Many challenges make server authentication difficult. There is not an obvious name by which to identify a tunnel method server. It is not obvious where in the tunnel server certificate the name should be found. One particularly problematic practice is to use a certificate that names the host on which the tunnel server runs. Given such a name, it is very difficult for a peer to understand whether that server is intended to be a tunnel method server for the realm.

It’s not clear what trust anchors to use for tunnel servers. Using commercial Certificate Authorities (CAs) is probably undesirable because tunnel servers often operate in a closed community and are often provisioned with certificates issued by that community. Using commercial CAs can be particularly problematic with peers that support hostnames in certificates. Then anyone who can obtain a certificate for any host in the domain being contacted can impersonate a tunnel server.

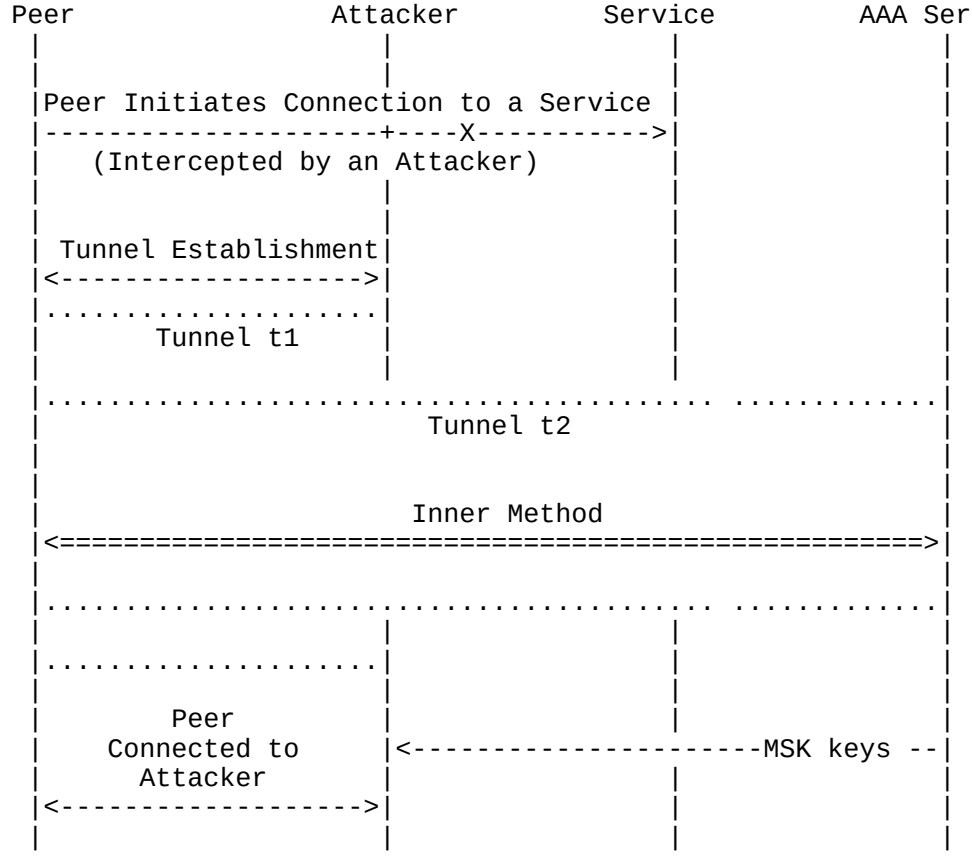
These difficulties lead to poor deployment of good certificate validation. Many peers make it easy to disable certificate validation. Other peers validate back to trust anchors but do not check names of certificates. What name types are supported and what configuration is easy to perform depend significantly on the peer in question.

Specifications also make the problem worse. For example, [RFC5281] indicates that the only impact of failing to perform certificate validation is that the inner method can be attacked. Administrators and implementors believing this claim may believe that protection from passive attacks is sufficient.

In addition, some deployments such as provisioning or strong inner methods are designed to work without certificate validation. Section 3.9 of the tunnel requirements document [RFC6678] discusses this requirement.

3.2.2. Server Policy

Server policy can potentially prevent the second condition (attacker being able to respond to inner authentication) from being possible. If the server only performs a particular inner authentication within a tunnel, then the attacker cannot gain a response to the inner authentication without there being such a tunnel. The attacker may be able to add a second layer of tunnels; see Figure 3. The inner tunnel may limit the attacker's capabilities; for example, if channel binding is performed over tunnel t2 in the figure, then an attacker cannot observe or influence it.

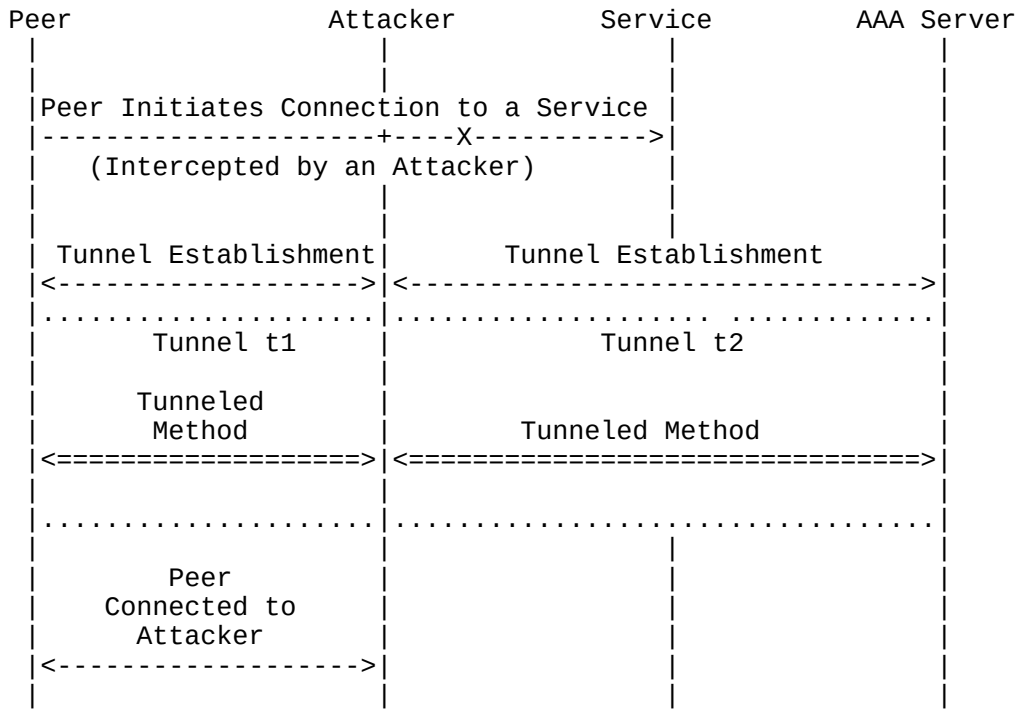


A tunnel t1 from the peer to the attacker contains a tunnel t2 from the peer to the home EAP server. Inside tunnel t2 is an inner authentication.

Figure 3: Multiple Layered Tunnels

Peer policy can be combined with this server policy to help prevent conditions 1 (attacker can establish a tunnel the peer will use) and 2 (attacker can respond to inner authentication). If the peer requires exactly one tunnel of a particular type and the EAP server only performs inner authentication over a tunnel of this type, then the attacker cannot establish tunnel t1 in the figure above. Configuring this peer policy may be more challenging than configuring policy on the EAP server.

An attacker may be able to mount a more traditional man-in-the-middle attack in this instance; see Figure 4. This policy on the peer and EAP server combined with a tunnel method that supports cryptographic binding will allow the EAP server to detect the attacker. This means the attacker cannot act as a legitimate NAS and, in particular, does not obtain the MSK. So, if the tunnel between the attacker and peer also requires cryptographic binding and if the cryptographic binding requires both the EAP server and peer to prove knowledge of the inner MSK, then the authentication will fail. If cryptographic binding is not performed, then this attack may succeed.



A tunnel t1 extends from the peer to the attacker. A tunnel t2 extends from the attacker to the home EAP server. An inner EAP authentication is forwarded unmodified by the attacker from tunnel t1 to tunnel t2. The attacker can observe this inner authentication.

Figure 4: A Traditional Man-in-the-Middle Attack

Cryptographic binding is only a valuable component of a defense if the inner authentication is a key-deriving EAP method. Most tunnel methods also support non-EAP inner authentication such as Microsoft CHAP version 2 [RFC2759]. This may undermine cryptographic binding in a number of ways. An attacker may be able to convert an EAP method into a compatible non-EAP form of the same credential to suppress cryptographic binding. In addition, an inner authentication may be available through an entirely different means. For example, a Lightweight Directory Access Protocol [RFC4510] or other directory server may provide an attacker a way to get challenges and provide responses for an authentication mechanism entirely outside of the AAA/EAP context. An attacker with this capability may be able to get around server policy requiring an inner authentication be used only in a given type of tunnel.

To recap, the following policy conditions appear sufficient to prevent a server insertion attack:

1. Peer and EAP server require a particular inner EAP method used within a particular tunnel method.

2. The inner EAP method's authentication is only available within the tunnel and through no other means including non-EAP means.

3. The inner EAP method produces a key.

4. The tunnel method uses cryptographic binding and the peer requires the other end of the tunnel to prove knowledge of the inner MSK.

3.2.3. Existing Cryptographic Binding

The most advanced examples of cryptographic binding today work at two levels. First, the server and peer prove to each other knowledge of the inner MSK. Then, the inner MSK is combined with some outer key material to form the tunnel's EAP keys. This is sufficient to detect an inserted server or peer provided that the attacker does not learn the inner MSK. This seems sufficient to defend against attackers who cannot act as a legitimate NAS.

The definition of cryptographic binding in [RFC3748] does not require these steps. To meet that definition, it would be sufficient for a peer to prove knowledge of the inner key to the EAP server. This would open some additional attacks. For example, by indicating success, an attacker might be able to mask a cryptographic binding failure. The peer is unlikely to be able to detect the failure, especially if only the tunnel key material is used for the final keys.

As discussed in the previous section, cryptographic binding is only effective when the inner method is EAP.

3.2.4. Introducing EMSK-Based Cryptographic Binding

Cryptographic binding can be strengthened when the inner EAP method supports an Extended Master Session Key (EMSK). The EMSK is never disclosed to any party other than the EAP server or peer, so even a legitimate NAS cannot learn the EMSK. So, if the same techniques currently applied to the inner MSK are applied to the inner EMSK, then condition 3 (completing tunnel authentication) will not hold because the attacker cannot complete this new form of cryptographic binding. This does not prevent the attacker from learning

confidential information such as a channel-binding request sent over the tunnel prior to cryptographic binding.

Obviously, as with all forms of cryptographic binding, cryptographic binding only works for key-deriving inner EAP methods. Also, some deployments (see Section 3.3) insert intermediates between the peer and the EAP server. EMSK-based cryptographic binding is incompatible with these deployments because the intermediate cannot learn the EMSK.

Formally, EMSK-based cryptographic binding is a security claim for EAP tunnel methods that holds when:

1. The peer proves to the server that the peer participating in any inner method is the same as the peer for the tunnel method.

2. The server proves to the peer that the server for any inner method is the same as the server for the tunnel method.

3. The MSK and EMSK for the tunnel depend on the MSK and EMSK of inner methods.

4. The peer MUST be able to force the authentication to fail if the peer is unable to confirm the identity of the server.

5. Proofs offered need to be secure even against attackers who know the inner method MSK.

If EMSK-based cryptographic binding is not an optional facility, it provides a strong defense against server insertion attacks and other tunnel man-in-the-middle (MITM) attacks for inner methods that provide an EMSK. The strength of the defense is dependent on the strength of the inner method. EMSK-based cryptographic binding MAY be provided as an optional facility. The value of EMSK-based cryptographic binding is reduced somewhat if it is an optional feature. It permits configurations where a peer uses other means to authenticate the server if the peer has sufficient information configured to validate the certificate and identity of an EAP server while using EMSK-based cryptographic binding for deployments where that is possible.

If EMSK-based cryptographic binding is an optional facility, the negotiation of whether to use it MUST be protected by the inner MSK or EMSK. Typically, the MSK will be used because the primary advantage of making EMSK-based cryptographic binding an optional facility is to permit intermediates who know only the MSK to decline to use EMSK-based cryptographic binding. The peer MUST have an

opportunity to fail the authentication after the server declines to use EMSK-based cryptographic binding.

3.2.5. Mix Key into Long-Term Credentials

Another defense against tunnel MITM attacks, potentially including server insertion attacks, is to use a different credential for tunneled methods from other authentications. This may prevent the second condition (attacker being able to respond to inner authentication) from taking place. For example, if key material from the tunnel is mixed into a shared secret or password that is the basis of the inner authentication, then the second condition will not hold unless the attacker already knows this shared secret. The advantage of this approach is that it seems to be the only way to strengthen non-EAP inner authentications within a tunnel.

There are several disadvantages. Choosing a function to mix the tunnel key material into the inner authentication will be very dependent on the inner authentication. In addition, this appears to involve a layering violation. However, exploring the possibility of providing a solution like this seems important because it can function for inner authentications where no other approach will work.

3.3. Intended Intermediates

Some deployments introduce a tunnel server separate from the EAP server; see [RFC5281] for an example of this style of deployment. The tunnel server is between the NAS and the EAP server. The only difference between such an intermediate and an attacker is that the intermediate provides some function valuable to the peer or EAP server and that the intermediate is trusted by the peer. If peers are configured with the necessary information to validate certificates of these intermediates and to confirm their identity, then tunnel MITM and inserted server attacks can be defended against. The intermediates need to be trusted with regard to channel binding and other services that the peer depends on.

Support for trusted intermediates is not a requirement according to the tunnel method requirements.

It seems reasonable to treat trusted intermediates as a special case if they are supported and to focus on the security of the case where there are not intermediates in the tunnel as the common case.

4. Recommendations

4.1. Mutual Cryptographic Binding

The Tunnel EAP method [TEAP] should gain support for EMSK-based cryptographic binding.

As channel-binding support is added to existing EAP methods, EMSK-based cryptographic binding or some other form of cryptographic binding that protects against server insertion should also be added to these methods. Mutual cryptographic binding may also be valuable when other services are added to EAP methods that may require a peer trust an EAP server.

4.2. State Tracking

Today, mutual authentication in EAP is thought of as a security claim about a method. However, in practice, it’s an attribute of a particular exchange. Mutual authentication can be obtained via checking certificates, through mutual cryptographic binding, or in very controlled cases through carefully crafted peer and server policy combined with existing cryptographic binding. Using services like channel binding that involve the peer trusting the EAP server should require mutual authentication be present in the session.

To accomplish this, implementations including channel binding or other peer services MUST track whether mutual authentication has happened. They SHOULD default to not permitting these peer services unless mutual authentication has happened. They SHOULD support a configuration where the peer fails to authenticate unless mutual authentication takes place. Discussion of whether this configuration should be recommended as a default is required.

The Tunnel EAP method [TEAP] should permit peers to force authentication failure if they are unable to perform mutual authentication. The protocol should permit this to be deferred until after mutual cryptographic binding is considered.

Services such as channel binding should be deferred until after cryptographic binding or mutual cryptographic binding.

An additional complication arises when a tunnel method authenticates multiple parties such as authenticating both the peer machine and the peer user to the EAP server. Depending on how mutual authentication is achieved, only some of these parties may have confidence in it. For example, if a strong shared secret is used to mutually authenticate the user and the EAP server, the machine may not have confidence that the EAP server is the authenticated party if the

machine cannot trust the user not to disclose the shared secret to an attacker. In these cases, the parties that have achieved mutual authentication need to be considered when evaluating whether to use peer services.

4.3. Certificate Naming

Work is required to promote interoperable deployment of server certificate validation by peers. A standard way to name EAP servers is required. Recommendations for what name forms peers should implement is required.

4.4. Inner Mixing

More consideration of the proposal to mix some key material into inner authentications is desired. Currently, the proposal is under-defined and fairly invasive. Are there versions of this proposal that would be valuable? Is there a way to view it as something more abstract so that it does not involve a combinatorial explosion as a result of considering specific tunnels and inner methods?

5. Survey of Tunnel Methods

5.1. Tunnel EAP (TEAP) Method

The Tunnel EAP method [TEAP] provides several features designed to limit man-in-the-middle vulnerabilities and provide a safe platform for peer services.

TEAP implementations support checking the Network Access Identifier (NAI) realm portion against a DNS subjectAlternativeName in the certificate of the TEAP server. TEAP supports EMSK-based cryptographic binding as a way to achieve mutual cryptographic binding. TEAP also supports MSK-based cryptographic binding for cases where the EMSK is not available; this cryptographic binding does not provide sufficient assurance for peer services. TEAP provides recommendations on conditions that need to be met prior to using peer services. These recommendations explicitly address when the MSK-based cryptographic binding is sufficient and when EMSK-based cryptographic binding is required. TEAP meets the recommendations for implementations outlined in this memo.

5.2. Flexible Authentication via Secure Tunneling (FAST)

EAP-FAST [RFC4851] provides MSK-based cryptographic binding. EAP-FAST requires that server certificates be validated. However, no guidance is given on how servers are named, so the specification does not provide enough guidance to interoperably enforce this requirement.

EAP-FAST does not support channel binding or other peer services, although the protocol is extensible and TLVs could be defined for peer services. If the certificates are actually validated and names checked, then EAP-FAST would provide security guarantees sufficient to use these peer services. However, the cryptographic binding in EAP-FAST is not strong enough to secure peer services if the server certificate is not validated and name checked.

5.3. EAP Tunneled Transport Layer Security (EAP-TTLS)

The EAP Tunneled Transport Layer Security Version 0 (EAP-TTLS) [RFC5281] does not support cryptographic binding. It also does not support peer services such as channel binding although they could be added using extensible AVPs.

EAP-TTLS recommends that implementations SHOULD validate certificates but gives no guidance on how to handle naming. Even if certificates are validated, EAP-TTLS is not generally suited to peer services. As an example, EAP-TTLS does not include protected result indication. So, an unprotected EAP success packet can end the authentication. In addition, it is difficult for a peer to request services such as channel binding because the server ends the authentication as soon as authentication is successful.

A variety of extensions, including EAP-TTLS version 1, improve some of these concerns. Specification and implementation issues complicate analysis of these extensions. As an example, most implementations can be tricked into using EAP-TTLS version 0.

6. Security Considerations

This memo examines the security considerations of providing new classes of service within EAP methods. Traditionally, the primary focus of EAP is authenticating the peer to the network. However, as the peer places trust in the EAP server, mutual authentication becomes more important. This memo examines the security of mutual authentication for EAP tunnel methods.

7. Acknowledgements

The authors would like to thank Alan DeKok for helping to explore these attacks. Alan focused the discussion on the importance of inner authentications that are not EAP and proposed mixing in key material as a way to resolve these authentications.

Jari Arkko provided a review of the attack and valuable context on past efforts in developing cryptographic binding.

Sam Hartman's and Margaret Wasserman's work on this memo is funded by Huawei.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

8.2. Informative References

[GSS-EAP] Hartman, S. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", Work in Progress, August 2012.

[PT-EAP] Cam-Winget, N. and P. Sangster, "PT-EAP: Posture Transport (PT) Protocol For EAP Tunnel Methods", Work in Progress, March 2013.

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000.

[RFC4510] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[RFC4851] Cam-Winget, N., McGrew, D., Salowey, J., and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, May 2007.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.

[RFC6677] Hartman, S., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, July 2012.

[RFC6678] Hoeper, K., Hanna, S., Zhou, H., and J. Salowey, "Requirements for a Tunnel-Based Extensible Authentication Protocol (EAP) Method", RFC 6678, July 2012.

[TEAP] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel EAP Method (TEAP) Version 1", Work in Progress, September 2013.

[TUNNEL-MITM] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunnelled Authentication Protocols", Cryptology ePrint Archive: Report 2002/163, November 2002.

Authors' Addresses

Sam Hartman
Painless Security

EMail: hartmans-ietf@mit.edu

Margaret Wasserman
Painless Security

EMail: mrw@painless-security.com
URI: http://www.painless-security.com/

Dacheng Zhang
Huawei

EMail: zhangdacheng@huawei.com