

EAP Working Group
INTERNET-DRAFT
Category: Informational
<[draft-ietf-eap-esteem-01.txt](#)>
[9](#) January 2003

B. Aboba, Editor
Microsoft

Eap STate machineE dEsign teaM (ESTEEM) Discussions

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC 2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet- Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document describes the deliberations of the EAP STate Machine Design TeaM (ESTEEM). This includes minutes of meetings, as well as position papers.

INTERNET-DRAFT

ESTEEM

9 January 2003

1. Key Issues for Resolution in RFC 2284bis

Nick Petroni

October 14, 2002

I see three major areas at first glance. This is not comprehensive, just off the top of my head.

a. EAP MUX/method and method/method interaction.

It seems most people have settled into the MUX paradigm and some of the details have been decided such as:

- MUX and not methods sending Success/Fail
- Methods cannot be interrupted (one method at a time)
- Identity is a special kind of method (can't be NAK'd, etc.)

but other details don't seem clearly defined. In particular

- How does information get passed from method to method? through the MUX? directly between methods? not at all? Since Methods can send Id REQ themselves, is it assumed all methods have access to this info?
- How does the MUX know a method has begun/finished? This is particularly important in terms of success/failure only being allowed at specific times. We have discussed authMethodSuccess and Fail indications, but I don't think these have been set in stone.
- How is order determined? Is this standardized, or just blanketed by policy? Do methods get to pick "I want to be followed by X and will only go after Y"? Is it enough to say something such as "MD5 before TLS" and

what does that mean for cases where the same auth method could be run twice, e.g. "MD5 TLS MD5". I don't see why one would want to do this, but should the protocol allow it? Since there could be multiple back-end servers involved, it seems more likely than I would hope. Is it significant which MD5 goes first?

- how are methods with specific requirements like being the last method run handled? Does the method tell the MUX this or is it part of policy? Packet filters have been discussed here, but how/when are these specified by the method?

b. Policy specification

- If order is part of policy, how is it specified?
- it seems that any method can update policy (as shown in the state machine draft) and only the MUX can determine if the whole policy has been satisfied. how does this work for encapsulated methods such as PEAP? do they have their own policy with the outer policy just

- requiring "PEAP with policy x" to be successful?
- can general policy rules such as "any protocol providing successful mutual authentication"? how are these specified?
 - are there limitations/specifications for how a method interacts with policy? For example, will statements such as "All methods must

INTERNET-DRAFT

ESTEEM

9 January 2003

update policy variable X" be added to the bis?

c. Error handling

Conversations have ranged from adding a new message to modifying an existing one or just punting on errors all together. This could go any direction, but all would affect the state machine.

[2.](#) Meeting minutes

[2.1.](#) September 25, 2002 Minutes

EAP Design Team Meeting Minutes

Scribe: Paul Congdon, HP

09/25/02

Yoshi's Paper

a. Issue 1 - How to handle notational conventions within state diagrams. There is also an issue relating to handling of the timing.

Suggest two solutions - use an already defined convention (e.g. IEEE 802.1X) or use other current and more formal conventions. Chuck believes current format is less formal and unclear on how steps within a state are executed and whether they are atomic or not. Paul suggests adding words from 802 documents that clarify. Other believe that real events should be noted in the transitions not just variables. Bernard raises a question about sub-machine interactions and the entrance points. Unclear what information is available to methods. For example, is information in an Identity exchange available to the method? What about NAK? Notification? Wording can be added to clarify entry points.

Do methods need to know about one another? If so, how is communication facilitated? We don't want to allow this to be an exercise for the reader. Perhaps RFC2284bis needs an abstract API to allow this? The concept of an EAP mux helps clarify how the state machines

interact.

Seems to be some agreement that we can slightly deviate from the [802](#) machine format, but keep the spirit intact such that it is obvious to the reader. This makes it easier to synch the EAP and 802.1X state machines.

b. Issue 2 - Identity exchange is optional, but the authenticator machine doesn't really support it well.

Some would prefer to see ID exchange as simply one of the methods. However, would the ID method ever occur without other methods invoked?

Esteem

Informational

[Page 3]

INTERNET-DRAFT

ESTEEM

9 January 2003

Not likely, but certainly multiple IDs exchanges could be possible. Methods can't interrupt other methods. There is an assumption that a variable exists that controls the authenticator to send or not the ID. This variable is not specified in 802.1X or the EAP state machine document.

>From the Radius server's point of view, the ID exchange starts out as a method, but RADIUS servers often receive the Access Request after the ID Request has been sent by the Authenticator, answered with an ID Response from the Supplicant, and the Identity placed in a User-Name attribute. So the RADIUS server typically does not request an Identity exchange.

There is a need for a needID variable that drives the backend state machine as well. It actually describes what the first step is for each method. How is this variable controlled when multiple methods are changed. For example, if needID is set then the Authenticator starts out by sending an Identity Request. If not, then it sends an Access-Request to the RADIUS server (which can't contain a User-Name attribute because the identity isn't known yet).

Consensus that a needID is needed, but unclear how it is controlled still. Still unclear if ID exchange should be thought of as a method. If it is a method, do we still need needID? If it is a method, it should fold into the current proposed machines without significant disruption. No conclusion yet if ID exchange is a method.

c. Issue 3 - Inconsistency when processing NAK.

Can't a NAK break the in-progress EAP method? Typically this

means the Supplicant doesn't want to run this method, so he NAKs with alternatives. It is up to the authenticator to decide what method will run next if any.

It appears that currently the NAK can be sent anytime, so perhaps it should be its own machine. It can be sent in the middle of a method to abort is, but some don't like this behavior, because it allows an attacker to stop an authentication in progress. The objective of the NAK message is to negotiate a method (it has also been used for negotiation of a sub-method within the method, but not clear it is appropriate for that), but should it be used to abort the method? The NAK as an abort doesn't fully describe why the abort is taking place, so perhaps an error or abort message needs to be created. There is no notification message from the peer to the authenticator. Sending anything in the middle of an exchange could be an issue anyway from a security point of view. If you simply stop responding to the exchange in the middle, it will timeout eventually and another method proposal will come along. This could be NAK'd at that time.

Esteem

Informational

[Page 4]

INTERNET-DRAFT

ESTEEM

9 January 2003

There seems to be consensus that NAK should only be used for method negotiation. For aborting we should do something different. Choices include keep it the same, timeout, new message, NAK with parameter, etc. These choices need to be explored further.

It was noted that the method really starts with the first message and the NAK is a response to the first message of the method, so reception of the NAK is really just a condition of the method. However, each method really needs to start out in a proposal phase where NAK is one possible response. How does the mux know what is going on here? Does the mux keep state, and know when the proposal phase has ended (after which NAK is no longer allowed)?

Another topic to discuss is acknowledgements of success and failure messages. This isn't possible in [RFC 2284](#), but perhaps a new message is needed?

Timer events in the state machines aren't described. This is tricky because user input changes all the timing.

Another issue raised is how much constraint can we put on all EAP methods? Is the EAP method state machine a black box or must it conform

to some more specific guidelines or some specific required states. For example, is there an abstract API that describes how an EAP method interacts with the mux and the EAP state machine? For example, are there variables set within the method that influence behavior of the mux? For example, the method might tell the mux whether authentication had succeeded or failed, etc.

Yoshihiro Ohba's Minutes

EAP Design Team Tele-conference Minutes

Date: 11am-12pm, September 25, 2002

Participants: Bernard Aboba, Jari Arkko, Paul Congdon*,
Robert Moskowitz, Yoshihiro Ohba*+, Bryan Payne,
Nick Petroni, John Vollbrecht

*Minutes takers

+Minutes taker of this memo

(Please point out if your name is missing or you were not participated in the conference call.)

Minutes:

All participants agreed on the proposed approach based on two position

Esteem

Informational

[Page 5]

INTERNET-DRAFT

ESTEEM

9 January 2003

papers, one for reviewing 802.11a (topic a) and the other for reviewing the EAP state machine draft (topic b). Both position papers include comparison with the RFC2284bis draft.

The following issues related to topic b were discussed (the issues are described in a write-up posted on the EAP design team mailing list by Yoshihiro Ohba):

- o Notational conventions for state diagrams
- o Optional identity support
- o NAK message handling

a. Notational conventions for state diagrams

Precise definition for notational conventions is required for the

state machine. It was agreed that using the conventions used in the IEEE 802.1X specification with allowing some augmentation is better for the following reasons:

- 802.1X conventions are precise
- By using 802.1X conventions, EAP state machine would be better aligned with 802.1X state machine.
- On the other hand, boolean variable-based event transition defined in 802.1X might not be useful if we consider simplicity for the state machine description. So some augmentation by allowing "real-event" based state transition would be better.

It is pointed out that in the Authenticator SM (State Machine) there are multiple entrance points from a sub-SM (i.e., Method Authenticator SM) and there is a question that the sub-SM behavior may depend on which entrance point is used. However, in the case of EAP, the sub-state machine behavior should not depend on where the transition came from, meaning that some additional convention for describing each method SM (e.g., every method SM starts from a single initialization state) might be necessary.

There is another issue on what information is available from each method SM (e.g., each method SM may require Identity information). This might be an issue of EAP abstract API.

b. Optional identity support

The RFC2284bis draft describes that Identity message exchange is an option, but the EAP state machine is not sufficient to support the option.

A flag variable "NeedID" is needed to indicate whether identity exchange is required or not. The variable is initialized within `initPolicy()`. It is also necessary to be able to change the value of this flag in the mid of authentication (at the beginning of each method) when multi-authentication is used, since some methods in a multi-authentication session may require identity exchange while others may not.

There is an open issue on whether the state for handling Identity exchange should be defined in a distinct method SM or in the multiplexer SMs (i.e., outside of any method SM, as described in the current state machine draft).

c. NAK message handling

There is an issue on where to define the state for handling NAK message.

A NAK message MUST NOT be used for "aborting" the current method in the mid of the method and should be used for the first EAP Request message of the current method only.

There are two approaches to achieve this:

- define a state for handling NAK message outside of each method SM, perhaps with defining a global variable that indicates whether the NAK message can be processed or not (the value of the variable can be set by each method SM).
- define a state for handling NAK message in each method SM.

d. Candidate issues to be discussed in the next tele-conference:

- o Remaining issues described in the write-up by Yoshihiro Ohba
- o Position paper from UMD
- o EAP multiplexer issues posted by John Vollbrecht
- o Any discussion on topic a.

[2.2.](#) October 2, 2002 Minutes

Minutes EAP State Machine Design Team

Date: 10/2/2002 11am-12:15 EDT

Yoshihiro Ohba's (yohba@tari.toshiba.com) position paper on the EAP state machine draft (topic b).

- * last week we discussed issues 1,2,3, this week begin with issue 4
- * Issue 4: Support for multiple authentication methods in a single session

Yoshihiro's Position: We should define policy parameters common to all EAP methods that provide for multiple authentication methods.

Discussion: the conversation ranged over a variety of subjects relating to the interaction between multiple methods, when notifications should be allowed, the nature of special "protected" methods, protected success and failure messages, and when success and failure are valid.

Some key points from the discussion:

- Some methods run as a protection of other methods. These methods appear to the MUX as a single method, but may have special requirements such as being the last method run and disregarding ALL unprotected messages, including Notifications. The discussion thus turned to when notifications should be allowed and whether methods are currently using notifications or whether these are messages that should be handled at the MUX layer. The decision was made to separate these issues from the current discussion.
 - * A survey would be sent to the EAP list to determine current usage of notification messages
 - * Position papers would be accepted on whether or not a method should be implementable such that it is only the last method and, if so, how that method interacts with the MUX. Some felt that it should be left to policy to disallow other methods after methods like PEAP and not the method itself.
- The group discussed the nature of method-specific success and failure and how they interact with EAP success/failure.
- The group discussed and agreed that Success messages should only be sent at the end of a sequence and not after each individual method succeeds. This implies that it is the MUX who controls sending success messages. This issue is revisited in issue 5 below.
- The group discussed the role of policy in deciding method order and concluded that policy can be responsible for

specifying method order, but that a default peer policy should be defined that provides for the allowed methods to be accepted in any order. There was additional discussion on the implications of peer and authenticator disagreeing about order since there is no synchronized schema between the two.

* Issue 5: Pass-through EAP authenticator state machine

Yoshihiro's position: Should define special state machine for pass-through

Discussion: The conversation lead primarily down two paths- (a) the differences in 1x and PPP with respect to last message delivery and the necessity to generalize for media independence. At this point the sending of a success message by the MUX was revisited considering both the 1x and PPP case

(b) the nature of Identity messages including the ability for either Authenticator or Auth Server to request ID, the possibility for different auth servers to be used depending on the ID response, and the authenticator's desire to always know auth replies for accounting/audit purposes.

Additional discussion brought up that a number of new mediums are being discussed, thereby illustrating the need for a single unified protocol definition

Next week will continue with the remaining item b position papers.

[2.3.](#) October 9, 2002 Minutes

Minutes of the EAP state machine Design Team conference call

Date: October 9th, 2002, 18:00-19:00 EET

a. Agenda

- Report on 802.1aa (Bob Moskowitz)
- University of Maryland position paper
- Bernard's position paper
- Jari's position paper

b. Report on 802.1aa

Bob was the only one present in the 802.1aa meeting, but he wasn't present in the conference call.

INTERNET-DRAFT

ESTEEM

9 January 2003

c. Jari's position paper

This position paper studied whether the proposed EAP state machine has (message, state) pairs for which no actions have been defined. Several such pairs were found. Some of these pair are missing on purpose, while others are either mistakes or under-specification. The paper classified the issues around the following main problems:

- When can Failure come?
- When can NAK come?
- When can Identity Request come?
- When can Notification come?
- Should 2nd, 3rd, ... internal method requests be modeled in the main state machine?
- Should we treat protocol errors in the state machine or not?

o Failure:

- The issue is whether peer should accept Failure only after a method execution, or also in the unauthenticated state before any methods.
- DECISION: Peer should accept Failure in unauthenticated state.
- There was a discussion of whether such failures can be based on the NAI given in an Identity Response. There are security issues around exposing valid / invalid user names. It was unclear whether there are similar issues in exposing domains for which the authenticator is unable to perform authentication.
- DECISION: We should document the security considerations about this, but not prevent / mandate any specific behavior.
- Bernard: Filters may prevent failure messages if the method is protected and wants a protected failure report. Methods may install such filters.

o NAK:

- We decided to talk about this in more general manner in the context of protocol errors.

o Identity request:

- The issue is whether Identity Request can be NAK'd, whether Identity Request can be repeated, whether it can come during the execution of a method or only in between.
- Also, can a method ask for an identity request to be sent?
- Can a method get the information from an identity response?
- John asked whether the state machine goes through the unauthenticated state as it moves from one method to another. The answer is yes.
- Bernard said that we should avoid having to give policy for EAP nodes; this is usually a sign of a lack of specification.

Esteem

Informational

[Page 10]

INTERNET-DRAFT

ESTEEM

9 January 2003

Sequences of methods, different combinations can create problems and lots of NAKking if policies dictate too much.

- Is identity a method? Can you NAK identity? RFC says no, and some implementations are known to not be able to process such NAKs.
- Bernard: RFC set Identity Requests to be optional, but 802.1x made it mandatory. This is a conflict. What if both sides don't want it?
- Bernard: Perhaps we should request 802.1x to make it optional also. This may not be easily adopted by 802.1aa, however. One issue is that they would like to have a solution from the IETF that works without configuration.
- Jari: There is a danger that if we don't allow NAKs, we will soon develop a convention where an empty NAI implies that the identity does not exist or we don't want to give it.
- DECISION: Identity request/response can only appear between methods
- DECISION: Our preference is that identity requests be optional. (And we are leaning towards making NAK disallowed.)
- DECISION: Work on the text on the list.
- DECISION: Talk with 802.1aa about the situation.

o Notification:

- DECISION: Discussion elsewhere appears to lead to allowing Notification in all states.
- DECISION: Can not be NAK'd.

o 2nd, 3rd messages

- John: Multiplexor model will make this clearer
- Can we execute methods in parallel, 1st message from 1st method, 1st message from 2nd method, ... 2nd message from the 1st method, 2nd message from the 2nd method, ...?
- DECISION: No. Methods must be done one at a time.
- There must be a concept of activating and deactivating

a method.

- Bernard: Acceptance locks you to the method
- Jari: Do we need to model this in state machine?
- John: yes

o Protocol errors:

- The issue is whether the state machine should say something about what to do when invalid EAP messages arrive. This may be necessary for recovery etc. purposes.
- Bernard: There's certainly errors in individual methods.
- Not all protocol errors need the same treatment
- Bernard: Its a reasonable question to ask what to do in a protocol error situation, if the spec says something is that
- Should silent discard be used?
- We should avoid making Denial-of-Service attacks too easy
- John would like to have positive confirmation of a failure

rather than stubbornly trying and timing out later

- Servers could send a Notification
- There is a usability tradeoff: failure report quickly vs better Denial-of-Service resistance
- John: Could we send a NAK with a message?
- Bernard: Implementation backwards compatibility prevents NAK changes
- Is there a need for a new message?
- Jari: We have the following options to deal with unexpected protocol messages:
 - 1) Silent discard and wait for proper message; this often leads to a timeout if the discarded message really was from the authenticator.
 - 2) Immediately abort everything and exit from EAP. This provides a quick indication to the user, but is very fragile wrt Denial-of-Service attacks.
 - 3) Provide a new abort message with support for authenticating that the other side really sent the offending message. It is unclear how complicated this is.
 - 4) A new notification message could be design for peer=>authenticator direction. This would allow implementations to report problems, without making the protocol very vulnerable to Denial-of-Service.
- DECISION: We need separate discussion on errors.
- Jari: I will write something about it.

d. Concluding remarks

- Other two position papers will be dealt with later.
- John: We've made a lot of progress lately in the issues. The main remaining issues are policy issues and dealing with protocol errors.
- Jari: We will continue with the conference calls, but probably need also to do e-mail reviews of the position papers.

2.4. October 23, 2002 Minutes

Present: Bob Moskowitz, Nick Petroni, Jari Arkko, Yoshihiro Ohba, Glen Zorn, John Vollbrecht, Paul Congdon, Joseph Salowey

Scribe: Jari Arkko

1. John's position paper.

- Was submitted, but bounced the list. He started to do state tables which assumes the methods can also provide information to the EAP mux. Clearly needs more work, but he'd be happy to pursue it.
- There's a couple of main things:
 - It would be possible to map this back to the states

Esteem

Informational

[Page 12]

INTERNET-DRAFT

ESTEEM

9 January 2003

that are in the state machine now. Though it doesn't make any distinction about methods, even identity is seen as a method.

- Assumes policing. Some of the actions assume that policies will dictate certain things.
- Starts to imply things about the API.
- Bernard: This is fairly complete.
- Bernard: There are two cases when you receive method X request. One, we are done and should change state. Two, we need further requests. In addition, there are error situations.
- Bernard: If within a method, you receive another method, is this NAK'd or silently discarded.
- DECISION: NAK is better than silent discard.

- Question: Is a failure of a method terminal?
We could either terminate EAP, or allow continuation with another method per policy?
- John thinks we should allow the latter.
- Jari thinks we should not.

- Question: is the treatment of final success or fail dependent on whether the previous method failed or succeeded?
- Proposal: We need a new state for this.

- Question: is there a global state, what has happened on all methods, useful maybe for re-authentication? How can re-authentication
- Answer: Good question.

- Jari: Why is not authenticated vs. no current methods there?
- Jari: Reformulated question: is Success / Fail illegal at the beginning?
- Bernard: Maybe this is policy?
- Jari: If we can avoid policy, we should.

2. Moving Forward

o Drafts:

- Bernard: I encourage John to write up this state machine with this draft.
- Jari: Should we have an individual draft by

- John, or merge drafts to WG draft status.
- DECISION: Too early now.
- Bernard: It would be better if Brian and Nick could do another revision of their draft as well.

- DECISION: Priorities are: 1) follow decisions 2) be complete 3) follow format.
- Jari: Need to capture current decisions. Bernard: We are tracking the issues.

- Bernard: One of the open issues is described above

- Yoshihiro: We also need to consider the pass-through state machine.
- Jari: That can also be done as a separate draft.
- DECISION: By the I-D deadlines, we will have the following:
 - 2-3 state machines
 - [rfc2248](#) updated to -07
 - some solved issues
 - many unsolved issues
 - position papers and minutes will also be published
- o IETF-55 and where do we go from there
 - We will have a total of 4.5 hours and 2 slots.
 - On the first day, state machine is important and the issues are important
 - Second day is still unclear
 - What kind of method discussion we can have is under discussion.
 - John: Method discussion is key for EAP.
 - Bernard: We need to try to discuss issues in the methods, see if they are affected by state machine modifications etc.
 - Process for evaluating methods.
 - Drafts solicited. Luca, Glen are writing something about this.
 - Glen: Cart is pushing the horse. Compatibility with 802.1x is not so important. Bernard: Tony Jeffrey views the 802 state machine as a way to shuffle EAP packets around. The job of the IETF state machine is to decide what to do with the packets. Because the IETF state machine is link layer independent, it is at a higher abstraction layer. We will use similar format, but it is not the intention to force this state machine to comply with 802 state machine.
- o Continue with dt meetings after initial I-D deadline?

- John: Not available next week.
- Jari: We should continue.
- DECISION: Meetings continue and next week we will go over the open issues.

2.5. October 30, 2002 Minutes

EAP Design Team meeting, Wednesday October 30th, 2002.

Minutes taker: Jari Arkko

o Agenda

Discussion of unresolved EAP issues:

<http://www.drizzle.com/~aboba/EAP/eapissues.html>

Discussion of cryptographic binding problems:

<http://www.drizzle.com/~aboba/EAP/draft-puthenkulam-eap-binding-01.txt>

<http://www.saunalahti.fi/~asokan/research/mitm.html>

Latest RFC 2284bis draft:

<http://www.ietf.org/internet-drafts/draft-ietf-pppext-rfc2284bis-07.txt>

o Issue 2: Alternative indications

- Should these be kept or not?
- Are protected success and failure a form of alternative indications
- Can the 802.1X messages be considered alternative indications
- Proposal: rate of loss is low enough, not a problem. Bernard: But PANA folks are expecting to run this on GPRS!
- Question: what is the right thing to do from a security point of view? Bob: we should require a success. Jari: depends on whether the indication is protected. Bernard proposes that if a protected indication is available, no alternative indications should be listened.
- Is link up and down an alternative indication? Bernard thinks its a different issue. In PPP, you would get back to LCP.
- EAPOL logoff, is it an indication or not? What if there's link layer security? Will it help?
- Bernard: We probably need to listen to these alternative forms of failure indication. This will open a DoS attack. Jari: The

question is if this makes the situation any worse, there may be other equivalent attacks already.

- Bernard: I don't like the argument but I can't argue much about it. Let's look at EAPOL logoff as an example. Attacker sends unencrypted EAPOL logoff messages to a multicast address. The access point will have to relay those to all devices on the wireless media, or even to the wired media. Debate whether peer will act on this? Bernard: it doesn't matter, we can get mac addresses and send unicast messages.
- Proposal (Bernard): Lots of ways to do these DoS attacks. Doesn't seem possible to ignore these indications.
- Bob: we will see link ups and downs all the time anyway, we need to ignore them in case the link comes back online soon. Bernard: Agree.
- Bernard: This is like IPsec and ICMP; most implementations don't believe ICMP but rather wait for an IKE timeout.
- Jari: the action depends on link layer, e.g. l2tp vs. physical link. Bob: this should go to the link layer specific specs.
- DECISION Bernard & Jari: if an authenticated indication exists, should not believe alternative indications (unless have to, like the link really went away). In the absence of authenticated indication, alternative failure indications may be accepted if the lower layer specs say so. Success indication should not be believed.
- Jari: I'm still unsure about this. What if I have successfully and mutually authenticated the other side? Why would I not believe an alternative success indication. Jari promises to write something about this later.

o Issue 7: How to run multiple methods one after another

- Bob: What would a non-supported peer do if it get another method and was expecting success or failure? Bernard: This ties to the last issue. If it gets a packet it can't make any assumption about the state. In Windows, authentication will time out. In a later version with support, how would you know if the peer supports sequences or not? Invent a new method to signal that you support them or have it written in the specification that the methods always support sequencing. Or negotiate? Bernard doesn't want to add a round-trip for negotiation.

INTERNET-DRAFT

ESTEEM

9 January 2003

- Bernard: We also need to discuss whether the end points have to be the same all the time.
- Questions: 1) How to do sequences? 2) how to prevent mitm attacks?
- Question: Could we add a continuation flag to the header in a backwards compatible way? (Bernard go look at the header.)
- Question: Can we send a NAK? Bob: In the Windows version it just sits there and can't even NAK after having executed a method. So we can't rely on NAK. The same applies to a new Command. So the end result is in any case a timeout. The good news is that all the latency is at the end.
- Yoshihiro says that if this results in a failure it is not a problem. This is because if the authenticator wanted to do multiple methods, the authentication will fail anyway.
- Bernard: Can we change the identity request to specify if the peer can do multiple methods? Or we could have this as a part of the user data base.
- DECISION (Jari): Its better to take the timeout at the end for the people who will fail, than spend negotiation time for everyone at the beginning.

o Issue 10: Lack of authenticated success and failure indications

- Bob: Security people want to have one point where success or failure is indicated.
- Bernard: Can we require authenticated indications in some situations, such as for sequences or tunnels? Probably we can't mandate it for existing clients and the basic EAP situation.
- Bernard: Can't have authentication until keys are generated. Bob: We can always do a Chap-like request-response which authenticates an indication.

We could have an acknowledge-accept approach, and have it

mandatory for some new places where EAP is used, such as GPRS or PANA EAP-over-IP.

- Jari: One idea is that if someone supports sequences, they could use a protected success -method as the last one in the sequence.
- Question: Is it sufficient to protect tunneled

Esteem

Informational

[Page 17]

INTERNET-DRAFT

ESTEEM

9 January 2003

indications, or do we really have to support also secure indications in sequences? (No opinion)

- DECISION: We can't put this in situations where there are no sequences or tunnels. The rest of the discussion has to be taken in IETF #55.

o Issue 13: Identifier usage not specified?

- Can we specify it to start from 0? It appears more important to specify how it increases and how ordering / duplicates are detected.
- Seems like the existing text is quite appropriate. Does not mandate too much, but specifies enough.
- DECISION: Existing text is OK for #13 & #18.

o Issue 23: Contents of identity request payload

- The problem is if we have enough information in Identity Request to be able to answer? In most cases we have some lower layer information, such as the dialed phone number that tells you which identity to use. Do we need something like this in the Identity Request message?
- Bernard: Should the authenticator tell its NAI? Jari: Isn't this a chicken and egg problem, how can the authenticator know what to tell before it knows from which roaming ISP this user is from? Bernard: There's methods out-of-band to EAP for knowing e.g. that we clicked the AOL icon so we are looking to give AOL identity.
- Bob: I'm working on an extension to give a list of NAIs in the

identity request (or in the response). The request contents are treated just as an advertisement. Bernard: This sounds good.

o Next meeting

- We will have a meeting next week (the last meeting before IEEE and IETF)

2.6. December 11, 2002 Minutes

Date: Wednesday, December 11, 2002

Time: 8 AM PDT

Present: Bernard Aboba, Jari Arkko, Bob Moskowitz, John Vollbrecht, Paul Congdon, Glen Zorn, ?

Scribe: Jari Arkko

Esteem

Informational

[Page 18]

INTERNET-DRAFT

ESTEEM

9 January 2003

1. Preliminary discussion

This was the first meeting after IETF 55.

Based on looking at the state machine drafts, they look good but not all design team decisions are yet adopted in them. The two drafts are being merged into one and some more issues are resolved in them at the same time.

IEEE 802.1aa ballot may affect state machine. We encourage people to read the document and comment.

2. Discussion of EAP issues

<http://www.drizzle.com/~aboba/EAP/eapissues.html>

- 43: Security properties

Jari: Worry about requiring proofs. Bernard: Yes, it may not be desirable. DECISION: We should encourage people to include references to proofs and other analysis, but not require proofs. Jari to write alternative text.

Bernard: Is the text on dictionary attack sufficient? Glen: Does not sound too good yet. Bernard: Even liveness doesn't prove resistance. Glen: Last part makes sense. Jari: I would just put

in that the spec must describe whether you are vulnerable to dictionary attacks. Glen: There is a difference between dictionary attack to a weak password vs. a method. DECISION: The following text can be used: Assuming there is a weak password in the secret, does the method allow an attack better than brute force? John: There is a further distinction between off-line and on-line attacks. DECISION: Let's not describe different variants of dictionary attacks, instead reference text books on the subject.

Key strength determination: How do we count key strength, do we count clear text nonces during EAP TLS negotiation to the entropy? There is some counter-intuitive input on this subject from known cryptographers. It is unclear if adding a PRF function will create additional entropy or not. DECISION: Bernard will ask the cfrg IRTF group what a good definition of key strength is, but we will not give a full definition in the specification.

- 2 Alternative indications

Bernard has new text on the issue site. Question: If after

receiving a protected success indication within a method, do you accept unprotected failure after the method has completed?

Bernard: The current text prevents you only from spoofing a success. If the method has not completed, you will throw both failure and success indications away. How all this is handled must be described in the state machines.

Also, within a tunnel are the state machines separate or joined?

Bernard: We should think them as a single state machine. Bernard: There are two issues here: whether the methods are completed or not, and whether the methods have sent their success/failure indications to the other side.

DECISION: If the method hasn't told the EAP mux layer that its done, the failure/success indications will be thrown away. Bernard: A method will just tell the mux when it is done. It may not know exactly about its success, but it indicates that it has completed successfully. If a method says success (e.g. within PEAP) and after that an unprotected failure

comes, what then? Jari: It is enough with the current text and protecting the success. Protecting the failure may not make sense as there are other DoS attacks in any case, e.g., on the methods.

Bernard: Is the primitive for completion just for my own success, or do we indicate also the other side's success? If we indicate the peer's success, then we could avoid spoofed failures. Jari: However, if we are running a sequence of methods, then it is no longer clear when a failure can be accepted. One possible semantics is that the protected indication applies only to what follows the protected method immediately -- but not what happens after the method. However, an attacker who can inject messages can always cause DoS by sending random messages, sending only the first message of a method etc.

DECISION: We will provide an indication from a method for both our own completion as well as the peer's completion.

- 25: Spoofing and duplicate detection

This attack is like attacking TLS TCP, but easier since the sequence number space is smaller in EAP.

Is there a difference in EAP methods as to how they want to receive their messages, either letting the Mux handle the sequencing or doing it by themselves using some kind of integrity protection to avoid these attacks. This is analogous

to the TLS over TCP vs. IKE situation.

Should we have a primitive from the method to the mux which says 'take me back to the previous sequence number'? This is also related to retransmission of broken packets. DECISION: We should have a primitive for tossing out messages that fail integrity protection.

There's also a timing issue: What if the real message arrives during the integrity verification? DECISION: The state machine must act in a lock-step; the EAP mux will not handle any new messages before the method processing has indicates that the

previous message has been either processed or thrown away.

- 41: NAK of extended types

This is has proven to be complicated if you mix regular and extended types. Bernard proposes that we only allow one kind of types in NAK.DECISION: We will allow NAK of 1 or more regular types OR a NAK of one vendor.

[2.7.](#) December 18, 2002 Minutes

EAP Design Team Notes 12/18/02 Scribe: Paul Congdon

802.1aa discussion

- [1] In Paul's ballot comments for 802.1aa/D4.1 he brought up some issues between the interface of 802.1X and the EAP layer. There have been many new checks put into the EAP layer and subsequent EAP Methods that may cause silent discards or state changes the 802.1X layer may need to know about. The current 802.1aa draft always assumes it sends a response to every request, so this doesn't support silent discards. The two machines need to be looked at side-by-side to assure a proper interface exists for the expected behavior.
- [2] It was agreed to prepare a reconciliation between the two layers and present at the 802.1aa interim meeting in Vancouver (1/6 - 1/10). Paul to send mail to Tony to request a specific time for the AA meeting. John and Bernard to suggest times when they could possibly attend the 802.1aa meeting to schedule this particular aspect.
- [3] Discussing the interface between media layers and EAP layers. It should be simple handshakes and should avoid putting media into particular modes or states if possible. Something that simply acknowledges the receipt of EAP frames from the media layer.

Esteem

Informational

[Page 21]

INTERNET-DRAFT

ESTEEM

9 January 2003

EAP state machine discussions

- [1] We need to consider EAP DoS attacks that can come from the wired side as well now that pre-auth is part of 802.11i. With Pre-

authentication, an attacker can authenticate to an AP, and then attack another AP within the same ESS. This means that EAP attacks do not have to be local, as would a DoS attack on an 802.11 cipher such as TKIP.

- [2] In the proposed state machine if any method in a sequence fails, the authentication fails.
- [3] There is a question about what happens on the peer when it perceives the authentication to have failed. For example, say the authenticator failed to authenticate to the peer, or it send a protected failure indication to the peer. Does the peer have to wait until receipt of an EAP-Failure? Or can it transition to another state immediately? What if the Authenticator thinks that the method is still ongoing and keeps retransmitting? How does the Peer terminate the conversation? There is a need for the peer to be able to signal termination of the authentication. A way to know that the peer has "hung up", so to speak. This is media specific, since not all media may have this ability. For example, with PPP, an LCP-Terminate can be sent; in IEEE 802.11, you can send a Dissociate or De-authenticate. But what do you do on an Ethernet? Lower carrier?
- [4] Some methods may support protected success/failure indications. In such a method, the authenticator will tell the peer that authentication has failed, and perhaps the Peer can respond, ACK'ing that assessment. In this case, assuming the exchange is completed, then both sides are in synch as to the outcome.
- [5] Retransmissions need to be checked by the EAP layer and tossed if necessary. How will you know the frame is truly a retransmission? John: All the bits must be checked to know this and the EAP layer isn't currently doing that. You can't just check the identifier. Bernard: Do you really want to require a comparison between a received frame and previously received frames for that identifier? Wouldn't this be creating a significant amount of work that would make a DoS attack easier? Remember, the underlying media is assumed to have a CRC check -- both PPP and IEEE 802 have this. If the CRC check fails, then the EAP packet isn't even processed by the EAP layer, it is discarded.
- [6] One possible approach here is for the EAP layer to queue packets for the method and then pass queued Requests up to the method, one

at a time. If the EAP method replies with a Response, the EAP layer should empty the queue since there should not be any additional packets in flight. If the Authenticator retransmit timer is off, it may have retransmitted earlier before receiving a Response, but that is not of concern to the Peer - clearing those packets from the queue has no ill effects. The Peer sends its Response; if a retransmission comes in after that, it can respond again; if not, it can handle the next Request. Clearing the queue is a DoS attack optimization.

- [7] If the EAP method didn't like the request it got (presumably because it failed an integrity check) it would tell the EAP layer "packet not accepted" and then the EAP layer will hand the next packet in the queue to the EAP method. Alternatively, the method could abort because it doesn't know how to handle integrity check failures (EAP TLS).

- [9] This implies that the EAP layer only does some basic checks: given the state of the method, is the Peer accepting packets other than Requests? Is the Type field acceptable? Is the length of the packet appropriate? There is no duplicate checking per se - the EAP layer just maintains a queue that is cleared after a Response is sent. This seems consistent with the behavior desired by Token Card methods. It could take a while for a person to enter in a Response - and if there is a retransmission during that time, you don't want to prompt the user again. You let them type in their Response, send it, clear the queue and then see if a retransmission occurs after that.

- [10] Some methods may not be able to survive an integrity check failure. For example, Yoshi pointed out that TLS cannot survive a MAC failure, so that presumably EAP TLS cannot survive this either. Bernard agreed to check on this.

- [11] Notification is handled differently than other methods. A Notification Request can be sent at any time, a Response is sent automatically by the EAP layer, and there is no state change resulting from this (other than perhaps Identifier state). John V was under the impression that Notification Request cannot occur in the middle of the method, but this was agreed upon in a previous Design Team meeting. In the ESTEEM draft we found the previous discussion that indicates the Notification Requests can be sent at any time and can't be NAK'd. We will stick to this decision, and John agreed to update the state machine document to reflect this. On the peer side it is fairly easy to just respond to the Notify, but on the authenticator, you must process the Notification Request like any other packet, wait for a response and retransmit as necessary, etc.

INTERNET-DRAFT

ESTEEM

9 January 2003

[12] John would like to re-visit this issue before making the changes. Could we meet on the 27th in the early afternoon or the 30th (the following Monday). Paul will get a phone line for the 30th to close this issue unless Bernard can secure the Microsoft line.

[3.](#) Position papers

[3.1.](#) Issues with the EAP State Machine Paper

By Yoshihiro Ohba (yohba@tari.toshiba.com)

Date: September 23, 2002

Issue 1: Need for detailed notational conventions
for state diagrams.

I think more work on the notational conventions of the current EAP State Machine is needed. For example, there is an ambiguity about the timing when the action defined in each state is executed. There are two possibilities:

- a) The action defined in a state is executed immediately after the state machine enters the state.
- b) The action defined in a state is executed immediately before the state machine leaves the state (and after testing branching conditions.)

It seems that the EAP state machine document assumes case a), but it is not clear.

In addition, although a state machine has a sub-state machine, it is not clear whether a sub-state machine's behavior depends on the entrance point from which the sub-state machine is entered when there are multiple entrance points.

Discussion:

Is it better to use the same conventions defined in [section 8.5](#) "EAPOL state machines" of IEEE 802.1X specification (EAPOL state machines

always uses case a), in which changes in boolean state variables are used as state transition events? Or is it better to use and improve the current conventions?

Issue 2: Optional Identity support

Esteem

Informational

[Page 24]

INTERNET-DRAFT

ESTEEM

9 January 2003

The current EAP Authenticator State Machine is not sufficient to support optional Identity exchange.

Suggestion:

- a. Define a variable "needID" which is initialized within initPolicy().
- b. Add a text tag "isSet(needID)" to the arrow connecting "Unauthenticated" state and "Peer Identification" state.
- c. Add a text tag "! isSet(needID)" to the arrow connecting "Unauthenticated" state and "EAP Method Authenticator State Machine" state.

Issue 3: NAK message handling

In the current EAP Authenticator State Machine, a state transition from "Unauthenticated" state to "EAP Method Authenticator State Machine" occurs when the Authenticator receives a NAK message.

Since both NAK and Identity messages are Native EAP type, it is better to have a distinct state for handling NAK reception events like the one used for handling Identity message.

Suggestion:

- a. Remove the transition arrow connected between "Unauthenticated" state and "EAP Method Authenticator State Machine". Also remove the text tag "Rec(NAK)" associated with the arrow.
- b. Add a new state "NAKReceived" and connect it to other

states as follows (only the related states and transitions are shown for simplicity):

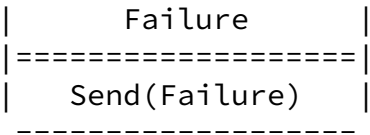
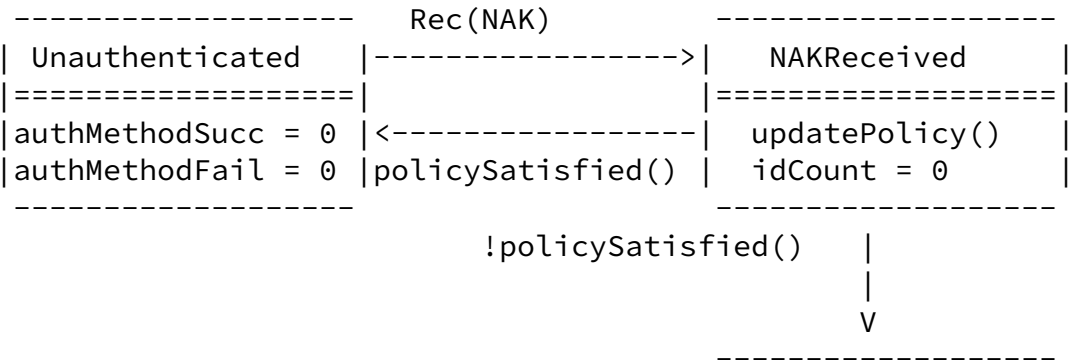


Figure 1. A Part of EAP Authenticator State Machine

Issue 4: Support for multiple authentication methods
in a single session

In order to support multiple authentication methods in a single session, it would be better to define policy parameters to support the functionality. Also, it would be better to define a primitive set of policy parameters which is common to all EAP methods. The policy parameters handling for multiple authentication methods can be included in the primitive policy set. Additional policy parameters can be defined in addition to the primitive policy set. All policy parameters including the primitive policy set are initialized when `initPolicy()` is called and dynamically modified when `updatePolicy()` is called.

Suggestion:

Define the following parameters for the primitive policy set:

EAPMultiAuthPolicy

A list of AuthPolicyElement's, where each AuthPolicyElement consists of the following entries:

- EAPMethodType

Contains an EAP Method Type used for the current round of authentication.

- SuccessPolicy

Contains a pointer to an AuthPolicyElement to be used for the next round of authentication when the current authentication round succeeds. If the value is null, the entire rounds of EAP authentication end with Success message.

- FailurePolicy:

Contains a pointer to an AuthPolicyElement to be used for the next round of authentication when the current authentication round fails. If the is null, the entire rounds of EAP authentication

end with Failure message.

CurrentAuthPolicy:

Contains a pointer to the AuthPolicyElement used for the current authentication round. The pointer is updated to the value contained in either SuccessPolicy or FailurePolicy of the current authentication round, depending on the result of the current authentication round. An optimization is possible when a NAK is received with non-null Type-Data. In that case, the CurrentAuthPolicy is updated by traversing the FailurePolicy side of each AuthPolicyElement until the pointer points to an AuthPolicyElement that has an EAPMethodType listed in the Type-Data of the NAK message.

Note:

The following variables can also be included in the primitive policy set.

idCountMax

This variable is already defined in the EAP state machine document.

needID

A flag to indicate whether Identity exchange is needed or not (see Issue 1).

Issue 5: Pass-Through EAP Authenticator State Machine

It would be useful to define a special EAP Method Authenticator State Machine used for pass-through mode Authenticator. By defining this, the same state machine definition can be used for Authenticator and authentication server, except that authentication server will not use the "Pass-Through" Method Authenticator State Machine.

The "Pass-Through" Method Authenticator State Machine is defined as follows:

Variables:

initialize

Esteem

Informational

[Page 27]

INTERNET-DRAFT

ESTEEM

9 January 2003

A variable that is set to TRUE when this state machine is created.

enterMethod

A variable that is assumed to be set to TRUE by the Authenticator State Machine when this state machine needs to be entered.

rxAuthReply

A variable that is set to TRUE when an authentication reply message is received from the backend authentication server.

rxAuthAccept

A variable that is set to TRUE when an authentication accept message from the backend authentication server.

rxAuthReject

A variable that is set to TRUE when an authentication reject message is received from the backend authentication server.

rxEAPResponse

A variable that is set to TRUE when an EAP Response message on this EAP method is received from the Peer.

error

A variable that is set to TRUE when an errornous event occurs, e.g., failure to send/receive message, etc. (Timeout event is excluded based on the convention described in the Introduction section of the state machine draft.)

Procedures:

txAuthRequest

send an authentication request message to the backend authentication server.

txEAPRequest

send an EAP Request message on this EAP method to the Peer.

States:

INITIALIZED

Esteem

Informational

[Page 28]

INTERNET-DRAFT

ESTEEM

9 January 2003

This state is entered when this state machine is created or the method-specific authentication ends.

AUTH_REQUEST_SENT

This state is entered when the Authenticator sent an authentication request to the backend authentication server and

is waiting a response from the authentication server.

EAP_REQUEST_SENT

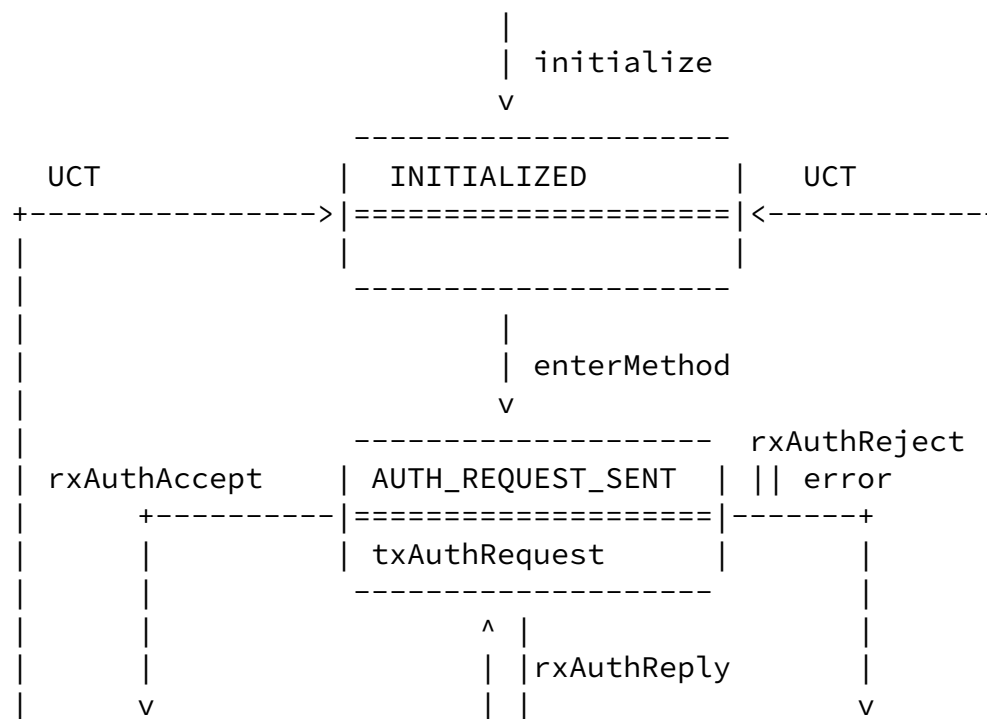
This state is entered when the Authenticator sent an EAP Request message on this authentication method to the Peer and is waiting an EAP Response message from the authentication server.

SUCCESS

This state is entered when the Authenticator receives an authentication accept message from the backend authentication server.

FAILURE

This state is entered when the Authenticator receives an authentication reject message from the backend authentication server.



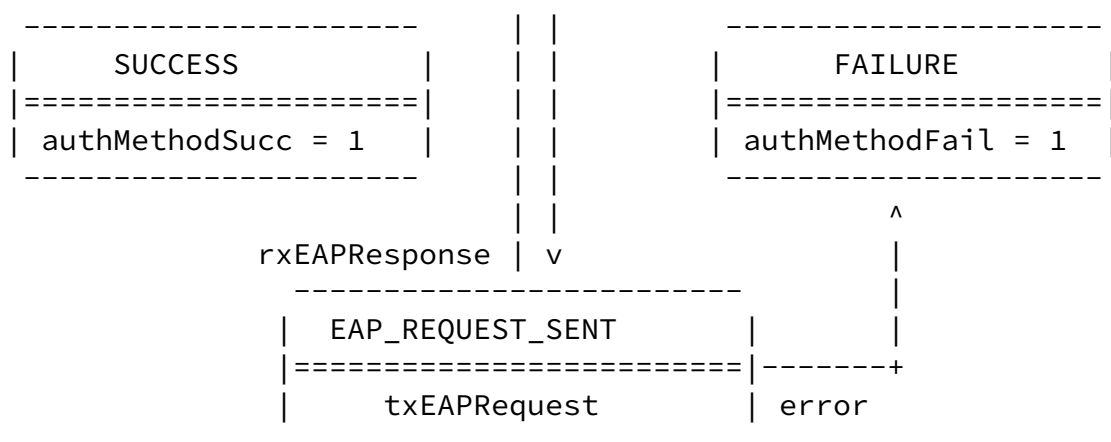


Figure 2. Pass-Through Method Authenticator State Machine

Note:

This diagram uses the notation conventions follows the ones used for the EAPOL state machines.

3.2. Comparison of EAP state machines with RFC 2284bis

Authors:

Bryan D. Payne (bdpayne@cs.umd.edu)

Nick L. Petroni, Jr. (npetroni@cs.umd.edu)

Date:

22 September 2002

References:

EAP State Machine Draft

<http://www.ietf.org/internet-drafts/draft-payne-eap-sm-00.txt>

RFC 2284bis

<http://www.ietf.org/internet-drafts/draft-ietf-pppext-rfc2284bis-06.txt>

INTRODUCTION

This paper outlines inconsistencies between the EAP state machine draft and the most recent revision of the EAP RFC. Suggestions are made for correcting any issues that are presented. In addition to inconsistencies, some typographic corrections and clarifications for the state machine draft are presented at the end of this paper.

INCONSISTENCIES

INTERNET-DRAFT

ESTEEM

9 January 2003

Below is a listing of the current known inconsistencies between the state machine draft and the EAP RFC:

Issue #1: The authenticator state machine is too restrictive in how an ID request may be used. Specifically, the state machines require each ID request to be followed by an Auth request, while the EAP RFC does not make this restriction.

Resolution #1: The authenticator state machine should be updated to transition from the Peer Identification state to the Unauthenticated state, instead of directly sending an Auth request.

Issue #2: The authenticator state machine does not provide any compatibility with RFC 2869bis. Specifically, there is no transition from the authenticator's unauthenticated state directly to the success and failure states due to receipt of a message from the back-end server.

Resolution #2: We do not propose a resolution for this issue at this time due to the complexity of the issues involved. However, we would like to leave this idea open to discussion.

Issue #3: The peer and authenticator state machines do not depict the re-authentication scenario. Instead, the state machines have a "dead end" at the authenticated states.

Resolution #3: Create a state transition from authenticated to initialization in both the peer and authenticator state machines. Indicate that this transition is to be used for re-authentication.

Issue #4: The state machine draft provides incorrect information regarding retransmission of messages (see Introduction, paragraph 4)

Resolution #4: Reword the discussion of retransmissions to indicate that the peer will never retransmit messages. In addition, note that the authenticator is allowed to retransmit up to a pre-specified number of times. Finally, note that both the peer and authenticator must return to the initialization state after "timing out" in any other state. Note that the definition of a time out is not provided in the EAP RFC. Therefore, this is implementation dependent, but should be a predefined amount of time with no activity.

Note that this is independent of the situation where the authenticator is permitted to retransmit messages after receiving invalid ID replies.

Also, note that the authenticator must never retransmit a success or failure message, per the RFC.

CLARIFICATIONS & TYPOGRAPHICAL CORRECTIONS

Esteem

Informational

[Page 31]

INTERNET-DRAFT

ESTEEM

9 January 2003

Below is a listing of topics that are viewed as correct in the current state machine draft, but that require additional clarification in the text of the draft to alleviate confusion or require a simple typographical correction.

Clarification #1: The state machines implicitly support the concept of a silent discard of the SUCCESS message. This is because the peer will not transition into an authenticated state unless its policy is satisfied and because messages that arrive at the peer, but are not handled by the current state must be dropped (see Introduction, paragraph 2).

While this support is present, it should be made more explicit due to the importance of the concept.

Clarification #2: In the peer state machine diagram, the lower arrow between the Peer Identification and the Unauthenticated states is pointing in the wrong direction.

Clarification #3: The definition of the authMethodSuccess variable in the peer state machine should be made more clear.

PROTOCOL LIMITATIONS

In the course of reviewing the state machine work, we were reminded of a protocol issue with EAP that should be understood by all. Unfortunately, this is not something that can be resolved by the state machine, so we only present it here for completeness.

The idea of doing a silent discard of success messages on the peer results in a bit of a problem. For example, let's say that the authenticator sends a success message, but the peer isn't ready to declare success. The authenticator will consider itself to be done, while the peer will have discarded the success message. Now what happens? The peer has no power (per the protocol) to send a request message...it must instead just wait for the authenticator to send one. So the peer has no way of indicating that it is unhappy with the situation.

There is a variable (authMethodSuccess) in the peer state machine to address this, but it is not an actual solution. It simply allows the peer to re-init if it isn't happy with the result of the auth method. However, since there is no way for the peer to tell the authenticator that it is unhappy, this could still leave the authenticator and the peer believing that the protocol has had different outcomes. This problem is touched on in EAP issue 10 ([1]).

Currently, we do not see any resolution to this problem. However, people should be aware that it exists.

Esteem

Informational

[Page 32]

INTERNET-DRAFT

ESTEEM

9 January 2003

SUMMARY

We have made several recommendations for corrections and clarifications to the state machine draft. We now encourage peer review of this work and are open to comments.

REFERENCES

- [1] <http://www.drizzle.com/~aboba/EAP/eapissues.html#Issue%2010>

3.3. Completeness review

EAP State Machine Completeness
Position Paper, Oct 1, 2002
Jari Arkko, Ericsson Research NomadicLab

A. INTRODUCTION

The completeness of the state machine from [draft-payne-eap-sm-00.txt](#) was analyzed. By completeness, we mean that the state machine contains a description of what to do for all possible events in all states.

Note that in some cases we may well ignore an event, but this should be done on purpose and we should document it clearly. If not, we often tend to forget some events that should have been handled with something else than silent discard. We have already debated a number of issues which are specific examples of the current under-specification that the original RFCs as well as the state machine proposals have, such as

when can NAKs come.

B. EVENTS

The events that can come to a peer are:

- ID Request
- Method Request (many types and subtypes)
- Notification
- Success
- Failure
- Protocol error (i.e. anything else, such as a NAK sent to the client -- this could be debatable whether we need to deal with this. But some protocols make a state transition e.g. close connection upon getting these.)

The events that can come to an authenticator are:

Esteem	Informational	[Page 33]
--------	---------------	-----------

INTERNET-DRAFT	ESTEEM	9 January 2003
----------------	--------	----------------

- ID response
- Methods response
- Notification ack
- NAK
- Protocol error

C. PEER STATE MACHINE

Some states in this state machine are just temporary transitions, such as the peer identification state, where we simply respond with an identity response and return back to unauthenticated state:

- Initialized
- Peer identification
- Invalid request
- Authenticated

The states on which some waiting of events occur are as follows:

- Unauthenticated
- Method state machine

We will study these two states in detail. We note that the Unauthenticated state does not handle the Failure, Notification, or protocol error events.

The method state machine state does not handle Identity request, Method request (perhaps this is thought to be on a lower level at the specific method state machine), Notification, and protocol error events.

D. AUTHENTICATOR STATE MACHINE

Again, the states which aren't real states in the usual definition of the term are:

- Initialization
- Failure
- Authenticated

The real states which we will study are:

- Unauthenticated
- Peer identification
- Method state machine

The Unauthenticated state does not handle Notification Response, NAK, and protocol error events (we suspect the right action in these cases would be silent discard).

The Peer identification state does not handle Method response, NAK, Notification, or protocol error events. Note that we have already discussed the situation where NAK might be used to signal that we don't want to give an identity.

The Method state machine state does not handle Notification response, Identity response, or protocol error events. It also doesn't really handle Method response event, but again we suspect this was expected to be done at the lower layer, in the specific method state machine.

E. FURTHER WORK

It is necessary to ensure that the state machines are complete and

we have knowingly decided to ignore the events that currently are ignored. We have already begun to debate some of the individual issues around this, such as when can NAKs come.

In the above analysis, we have not yet taken a final stand on whether the right action in some of these cases is something else than silent discard. If the right action is silent discard, then everything is currently OK.

It is also necessary to take a similar look at the 802.1aa D3 state machines.

[3.4.](#) When can packets be sent?

Bernard Aboba

Date: 10/2/02

When can a NAK be sent?

The authenticator proposes a method. The peer can accept the proposal (by replying with an EAP method of the same type), or NAK the proposal. Once the proposal is accepted, a NAK cannot be sent by the peer until a new method has been proposed. NAKs are never sent by the authenticator, nor are they sent by the peer in response to a method whose proposal has already been accepted.

When can a Notification be sent?

EAP Notification is **not** an error message, and so sending or receiving Notification Request/Responses does not change the state of the peer or authenticator. The peer **MUST** send a Notification Response to a Notification Request sent by the authenticator.

An EAP Notification Request may be sent at the behest of an EAP method running on the Authenticator, as part of the method. This implies that a Notification Request may only be sent after a method has been proposed by the authenticator and accepted by the peer.

Use of EAP Notification messages **MUST** be described as part of the method specification. Since the EAP Response does not contain any data, and no state change result from receipt of a Notification message, it should not be assumed that the

Notification Request message is processed by the EAP method on the peer; the Notification Response is sent automatically.

When can an Identity Request be sent and accepted?

The rules for sending an Identity Request are the same as with any other method. An Identity Request may be sent multiple times within a single EAP conversation, as is the case for any other method.

When are Success or Failure message sent?

EAP Success or Failure messages are sent by the authenticator EAP layer. When it completes, an EAP method indicates the result of the authentication to the EAP layer, by setting the Result=Success or Failure. If the method is the last in a sequence of methods, then the authenticator EAP layer will send the corresponding EAP Success or Failure message to the peer. If the method is not the final in the sequence, then the authenticator EAP layer will propose another EAP method to the peer.

Note that in order for a method to be usable within a sequence of methods, the final message sent within the method (prior to transmission of an EAP Failure or Success) MUST be an EAP Response. Otherwise, since EAP is an ACK/NAK protocol, there would be no way for the authenticator to send an EAP Request proposing another method.

When can a Success or Failure message be accepted?

EAP methods may include their own (protected) success and failure indications. Since EAP does not provide support for error messages, several methods have implemented method-specific error message capabilities. For example, TLS-based methods utilize TLS alerts, which are protected within the TLS channel in order to transmit error messages.

Similarly, methods also implement method-specific success indications. For example, a method supporting mutual authentication will expect the authenticator to complete authentication prior to completion of the method.

As a result, a method specification may indicate that the method only expects to receive clear text EAP Failure and Success messages under certain conditions. For example, a clear text Success message is only expected

once mutual authentication has been completed, or a clear text Failure

message is only expected after receipt of a protected failure indication such as a TLS alert.

These directives can be indicated via a filter interface between the method and the EAP layer. An EAP method can install a filter in the EAP layer, requesting it to discard EAP packets not meeting the filter criteria prior to completion of the EAP method. The operation of the filter MUST be described in detail within the EAP method specification, to enable interoperability. Filters installed by methods are automatically removed once the method completes.

For example, an EAP method can negotiate a key early on, and after that, may want to require that all EAP packets be protected using the derived key until a protected success/failure indication is received. To achieve this, the method on the peer may install the following filter within the EAP layer:

- (1) ACCEPT: Code = Request, Type=MyMethod
- (2) DENY: all

This will cause the peer EAP layer to silently discard messages of types Identity, Notification, Success and Failure until the filter is removed or the method completes. Once a protected Success indication is received (e.g. the authenticator completes mutual authentication), the filter list can be changed to:

- (1) ACCEPT: Code = Request, Type=MyMethod
- (2) ACCEPT: Code = Success
- (3) DENY: all

Similarly, once a protected failure message is received, a filter can be plumbed enabling the reception of an EAP Failure:

- (1) ACCEPT: Code = Request, Type=MyMethod
- (2) ACCEPT: Code = Failure
- (3) DENY: all

[3.5.](#) EAP method layer primitives

Communication between the Method and EAP layer

Bernard Aboba

October 15, 2002

One of the major issues that has arisen in the EAP Design Team discussion is the communication that occurs between EAP methods and the EAP layer, and what this enables.

EAP method Type = X	EAP method Type = Y	Native Methods (MD5)	EAP method Type = X	EAP method Type = Y	Native Methods (MD5)
!			^		
!			!		
!	EAP (NAK, Success, Failure, Notif., Id)		!	EAP (NAK, Success, Failure, Notif., Id)	
!			!		
!	Link Layer		!	Link Layer	
!			!		
!			!		

This primitive indicates to the EAP layer that the method considers itself to have completed EAP authentication, and if so, the result. This is important so that the EAP layer can know whether it is appropriate to accept or discard illegal messages. This includes Identity, NAK, Success or Failure messages sent in the middle of an EAP conversation, or messages of another EAP Type. The EAP-Method.COMPLETE primitive

also results in the removal of filters installed by the EAP method, if any.

INTERNET-DRAFT

ESTEEM

9 January 2003

d. EAP-Method.REQUEST_KEYS (Authenticator and Peer)

This primitive requests that the EAP layer provide the method with the keys derived by previous methods in the EAP conversation. This is important to enable cryptographic binding between methods in a sequence of methods or within a tunnel. This enables the endpoints to know that the entity in method #N is the same one that they were talking to in method #i where i=1,N. For example, if method #1 derives a key, then method #2 should demonstrate mutual possession of this key, in addition to deriving its own key, which subsequent methods could use to prove their binding. Without this binding, EAP tunneling or sequences of methods are subject to man-in-the-middle attack.

e. EAP-Method.PROVIDE_KEYS (Authenticator, Peer)

This primitive provides keying material from the EAP method to the EAP layer. This is needed in order to allow the AAA server to communicate keying material with the NAS device.

f. EAP-Method.INSTALL_FILTER (Authenticator, Peer)

This primitive requests that the EAP layer install a filter. This allows the EAP layer to reject additional packets in addition to the built-in filters. For example, the method could choose to reject notification messages while it is running, even though these don't change the state.

g. EAP-Method.REMOVE_FILTER (Authenticator, Peer)

This primitive requests that the EAP layer remove a filter. This allows the filter to be removed prior to completion of the EAP method. EAP filters are always removed on completion.

h. EAP-Method.SEND_PACKET (Authenticator, Peer)

This primitive requests that the EAP layer send an EAP packet.

i. EAP-Method.RECEIVE_PACKET (Authenticator, Peer)

This primitive requests that the EAP method receive an EAP packet from the EAP peer.

j. EAP-Method.SEND_NOTIFICATION (Authenticator)

This primitive requests that the EAP layer send a Notification request.

[3.6.](#) EAP State Machine Position paper

The following is my proposal for changes to the state machine to support enhanced protection against DOS attacks.

Esteem

Informational

[Page 39]

INTERNET-DRAFT

ESTEEM

9 January 2003

State Machine changes

This note describes changes to the Peer state machine. The major intent is to protect it from attacks that send EAP messages with the intent of creating a sequencing error. If this approach seems reasonable I can make corresponding changes to the authenticator state machine.

As a simplification of my earlier state machine I have also changed the state machine to assume that a method failure terminates the EAP exchange. Methods may sequence, and selection of the sequence may be negotiated with NAKs of requests, but once a method is negotiated it must succeed or the sequence fails and stops. Note: This puts a burden on EAP method implementers: a failure noted by the peer must be communicated to the authenticator, and the authenticator method must also signal failure.

The state machine assumes that the peer EAP Switch gets inputs from the remote Authenticator EAP Switch in the form of EAP Requests/Success/Failure messages. It gets inputs from EAP methods via an API which allows the method to pass EAP response and a signal about the state of the method.

As of the conversation at the last 12/11/2002 EAP Design Team call, my understanding is that it is worthwhile to expect additional information from the method. The method will provide the following signals (including 2 new signals as noted) to the EAP Switch when finished processing a Request.

[1.](#) Continuing - Method expects additional requests in order to successfully complete

[2.](#) Integrity-Check-Failure - Method did not recognize the Request. Treat

as a NoOp [new]

[2.](#) Done - Method is done. It does not expect more Requests for this method

[2.1](#) Method accepts authenticator

[2.2](#) Method fails authenticator

[2.3](#) Method accepts authenticator and authenticator accepts peer [new]

The reason for adding 2.3 is to encourage writing methods where each end knows that it is accepted by the other without having to send an EAP Success. The method can do this in a protected way, and if it does the information can be used to protect against DOS attacks. In the following state machine if a method signals the mutual acceptance (2.3) then it will not accept an EAP fail, and on timeout it assumes that the EAP success was lost in transit and acts accordingly.

Esteem

Informational

[Page 40]

INTERNET-DRAFT

ESTEEM

9 January 2003

Each of the signals noted above can be supported by a different state and each of the states has a set of "expected" messages and silently discards unexpected messages. This means that an unexpected request would be discarded rather than NAK'd. This is safer from DOS point of view, but perhaps less desirable if one wants a "positive indication of failure" in operation.

Retransmissions

In addition to the new signals we discussed dealing with retransmissions in the presence of attacks that insert random EAP requests. Note that the Peer does not retransmit, it has to sort out retransmitted requests from original or bogus requests. It's timer signal means it has waited for as long as it thinks reasonable for the authenticator to retry.

In what follows I assume that the EAP Switch will catch the "obvious" retransmissions. That is it will track the current method/seq and not present it twice. Since EAP is a REQ/REPL protocol it needs to be track that the req it just received is not a duplicate of one just processed. I assume this is done by the EAP Switch prior to passing it to the state machine.

Messages which seem valid to the EAP Switch may not seem valid to the method itself. The method is the authority and should do integrity checks on incoming messages. If a message fails an integrity check then

the method signals this failure to the Switch and the Switch treats this as a no-op. This is actually a problem because no-op means that the Switch must keep not only its current state but its previous state. Getting an integrity check message causes it to go back to the previous state.

Paul Congdon has suggested an alternative to the method proposed here. As I understand it, his suggestion is to leave all the handling of retransmissions to the method. This is an interesting approach and worth considering. The problem I see with it is that the method can be done and still have to deal with follow on messages. However, since the method must make this check anyway, it does not seem unreasonable to have it also deal with the how retransmissions. I have not worked out how this would work in this state machine, but it does not seem an insuperable problem - it is worth more discussion to work out which way is better.

To tie this all together, the following is a revised description of the Peer EAP Switch State Machine, [section 5](#) in the draft/internet-draft emailed to the group before the last IETF meeting.

Peer EAP State Machine

Esteem

Informational

[Page 41]

INTERNET-DRAFT

ESTEEM

9 January 2003

Peer EAP States

The peer EAP switch has the following states:

- | | |
|------------------------------|-----------------------------------|
| 1.inactive/not authenticated | - waiting for an EAP request |
| 2.active/waiting-method1 | - waiting for 1st method response |
| 3.active/waiting-methodc | - waiting for ongoing method resp |
| 4.active/method=x | - waiting for next EAP request |
| 5.active/method-ok | - waiting for method/Success/Fail |
| 6.active/method-succeeds | - waiting for method/Success |
| 7.active/failed method | - waiting for EAP Fail |
| 8.inactive/authenticated | - authenticated; reauth possible |

The typical sequence for a successful authentication is as follows. The Peer receives an EAP request from the Authenticator, going to state 2 and forwarding the request to the appropriate method. It receives a response from the method, going to state 4 and sending the EAP response to the Authenticator. It receives the next EAP Req from the

Authenticator and goes to state 3, forwarding the Req to the method. It receives a "final" response from the method and goes to state 5 (or 6) and sends the Response to the authenticator. It then receives an EAP Success from the authenticator and goes to state 8, authenticated.

Peer EAP Events

Events recognized by the peer EAP switch are listed below. Some Switch policy is implied in the events; e.g. EAP-Req/ok implies that the policy evaluated the Req as ok at this point.

>From Authenticator

- [1.](#) EAP-req/meth=x - EAP req for method x
- [2.](#) EAP-req/notok - req for method not allowed by policy
- [3.](#) EAP Success
- [4.](#) EAP- Fail

>From Method

- [5.](#) Method-resp.done.ok - peer accepts authenticator
- [6.](#) Method-resp.done.scs - peer and authenticator mutually ok
- [7.](#) Method-resp.done.fai - failure termination from method
- [8.](#) Method-resp-cont - continuing response from method
- [9.](#) Method-resp-IC - Integrity Check failed

Other

- [10.](#) Timeout

Peer EAP Actions

Esteem

Informational

[Page 42]

INTERNET-DRAFT

ESTEEM

9 January 2003

Actions initiated by the peer EAP layer are:

To Authenticator

- [1.](#) Send EAP-NAK
- [2.](#) Send EAP-Resp

To Method

- [3.](#) Forward EAP-req to method
- [4.](#) Send method terminate to method

To System

5. Signal accept or reject to system

Other

6. Silent Discard

EAP Peer State Table

State	Event	Next State	Action	Action2
inactv/unauth	EAP-rq/meth=x	actv/wtng-mth1	EAP-rq to mth	
	EAP-rq/notok	not-actv/auth	snd NAK	
	EAP-Scsess/ok	inactv/auth	Signal - acpt	
	EAP-Scsess/notok	not-actv/auth	Signal-rej	
actv/wtng-mth1	mth-rsp.done.ok	actv/no mth	snd EAP rsp	
	mth-rsp.done.fail	actv/failed-mth	snd EAP rsp	
	mth-rsp.done.sccs	actv/sccs-mth	snd EAP rsp	
	mth-rsp-cont	actv/mth=x	snd EAP rsp	
	mth-rsp-IC	prev-state		
	tmout	not-actv/auth	term to mth=x	
	*	actv/wtng-mth	silent discard	
actv/wtng-mthc	mth-rsp.done.ok	actv/no mth	snd EAP rsp	
	mth-rsp.done.fail	actv/failed-mth	snd EAP rsp	
	mth-rsp.done.sccs	actv/sccs-mth	snd EAP rsp	
	mth-rsp-cont	actv/mth=x	snd EAP rsp	
	mth-rsp-IC	prev-state		
	tmout	not-actv/auth	term to mth=x	
	*	actv/wtng-mth	silent discard	
actv/mth=x	EAP-rq/mth=x	actv/wtng-mth	EAP-rq to mth	
	EAP-rq/not mth=x	actv/mth=x	Silent Discard	
	EAP-Scsess	actv/mth=x	Silent Discard	

Esteem

Informational

[Page 43]

INTERNET-DRAFT

ESTEEM

9 January 2003

EAP-Fail	actv/mth=x	Silent Discard	
tmout	inactv/unauth	Signal-rej	term-mth
*	actv/mth=x	silent discard	

actv/mth-ok	EAP-rq/meth=x	actv/mth-ok	Silent Discard
	EAP-rq/notok	actv/mth-ok	Silent Discard
	EAP-Scsess	inactv/auth	Signal - acpt
	EAP-Fail	inactv/unauth	Signal-rej
	tmout	inactv/unauth	Signal-rej
	*	actv/no mth	Silent Discard
actv/sccs-mth	EAP-rq/meth=y	actv/wtg-mth1	EAP-rq to mth
	EAP-rq/notok	actv/mth-ok	snd NAK
	EAP-Scsess	inactv/auth	Signal - acpt
	EAP-Fail	inactv/unauth	Silent Discard
	tmout	inactv/auth	Signal-acpt
	*	actv/no mth	Silent Discard
actv/fail-mth	EAP-rq/mth=x	actv/failed-mth	snd NAK
	EAP-rq/notok	actv/failed-mth	snd NAK
	EAP-Fail	not-actv/auth	Signal-rej
	EAP-Scsess	actv/fail-mth	Silent discard
	tmout	inactv/auth	Signal-rej
	*	actv/failed-mth	silent discard
inactv/auth	EAP-rq/ok	actv/mth=x	EAP-rq to mth
	EAP-rq/notok	inactv/auth	snd NAK
	*	inactv/auth	silent discard

[4.](#) Surveys

[4.1.](#) NAK Survey

Date: Sun, 6 Oct 2002 02:37:28 -0700 (PDT)
 From: Bernard Aboba <aboba@internaut.com>
 To: eap@frascone.com
 Subject: NAK survey

During the last EAP design team meeting, there were also questions about the NAK Type (see issues #35 and 36).

Please answer the following questions relating to the operation of your implementation:

a. May the NAK be sent by the peer in response to any EAP request? For example, may it be sent in the middle of an EAP method? OR

- b. May the NAK be sent by the peer only in response to a method proposal (e.g. the first EAP Request for a given Type)?
- c. Does your implementation tolerate sending more than one Type within a NAK (e.g. may not recognize additional Types, but doesn't blow up)?
- d. Does your implementation permit a NAK to be sent in response to an Identity Request? If so, what does it do if a NAK is sent?

Date: Mon, 7 Oct 2002 07:13:34 -0400
From: James Carlson <james.d.carlson@east.sun.com>
To: Bernard Aboba <aboba@internaut.com>
Cc: eap@frascone.com
Subject: Re: [eap] NAK survey

Bernard Aboba writes:

- > During the last EAP design team meeting, there were also questions about
- > the NAK Type (see issues #35 and 36).
- >
- > Please answer the following questions relating to the operation of your
- > implementation:

<ftp://playground.sun.com/pub/eap/index.html>

- > a. May the NAK be sent by the peer in response to any EAP request? For
- > example, may it be sent in the middle of an EAP method? OR

Yes, it can be sent at any time.

- > b. May the NAK be sent by the peer only in response to a method proposal
- > (e.g. the first EAP Request for a given Type)?

No, at any time. I don't see how it could be otherwise -- for a multi-stage method, it's entirely possible to get part way down an authentication attempt only to discover that some required feature is missing in the peer and that some other mechanism is necessary.

- > c. Does your implementation tolerate sending more than one Type within a
- > NAK (e.g. may not recognize additional Types, but doesn't blow up)?

It ignores any additional Types included in a NAK and uses only the first one. (If *no* suggested Type is included with the message, then it logs this fact, and treats it as a failure of the current message.)

No, it doesn't "blow up," even though including multiple Types (or any other data) would be a violation of [RFC 2284](#). ;-}

> d. Does your implementation permit a NAK to be sent in response to an

Esteem

Informational

[Page 45]

INTERNET-DRAFT

ESTEEM

9 January 2003

> Identity Request? If so, what does it do if a NAK is sent?

If the identity of the peer was supplied by explicit configuration, then it accepts the NAK and continues with authentication using the suggested Type (if permitted).

If the identity was not supplied by explicit configuration, then it treats this case as authentication failure, sends EAP Failure, and shuts down the link.

There's one other case that your survey didn't cover (perhaps it's not a bone of contention?): what do you do when you get a NAK Type that isn't one that you want to deal with (either by administrative configuration or because you don't recognize it)? I examine my current state and ask for the "next" possible method on the list, returning to Identify if I fall off the end.

--

James Carlson, Solaris Networking	<james.d.carlson@east.sun.com>
SUN Microsystems / 1 Network Drive	71.234W Vox +1 781 442 2084
MS UBUR02-212 / Burlington MA 01803-2757	42.497N Fax +1 781 442 1677

Date: Mon, 7 Oct 2002 10:22:28 -0700
From: Sachin Sheth <sachins@windows.microsoft.com>
To: aboba@internaut.com
Subject: RE: NAK survey (fwd)

a. May the NAK be sent by the peer in response to any EAP request? For example, may it be sent in the middle of an EAP method? OR

EAP will send out only in the first packet after the EAP-Resp/Id. It will not be sent out at any other time.

b. May the NAK be sent by the peer only in response to a method proposal (e.g. the first EAP Request for a given Type)?

Yes

c. Does your implementation tolerate sending more than one Type within a

NAK (e.g. may not recognize additional Types, but doesn't blow up)?

There is no way to program this. EAP will only send out one EAP-type in the NAK packet.

d. Does your implementation permit a NAK to be sent in response to an Identity Request? If so, what does it do if a NAK is sent?

Esteem

Informational

[Page 46]

INTERNET-DRAFT

ESTEEM

9 January 2003

The NAK is sent out only after the EAP-Req/Identity is sent out.

Date: Mon, 7 Oct 2002 14:12:42 -0400
From: James Carlson <james.d.carlson@east.sun.com>
To: Bernard Aboba <aboba@internaut.com>
Cc: eap@frascone.com
Subject: Re: [eap] RE: NAK survey

Bernard Aboba writes:

> For what it's worth, here's what Microsoft Windows XP does:
>
> a. May the NAK be sent by the peer in response to any EAP request? For
> example, may it be sent in the middle of an EAP method? OR
>
> Windows XP peer will send a NAK only for the first packet after the
> EAP-Response/Identity is sent. It does not NAK an EAP-Request/Identity.

Interesting question here: was the question above about what you'd *accept* or about what you'd possibly *send*?

It looks like I read the question as the former (because "sent by the peer" implies "received by the local system" to me), and you might have read it as the latter. If it's intended as the latter, then I'd say this: I never send NAK in response to an Identity message, but I *do* send NAK in response to *any* message that is malformed for the method in use. In other words, if the method specifies a 16 octet message for the given message subtype, and you send me fewer than 16 octets, then I assume that you're running some new or different version of the protocol, and I send NAK to suggest a different authentication protocol since we're not going to converge.

--

James Carlson, Solaris Networking

<james.d.carlson@east.sun.com>

4.2. Notification Survey

Based on the survey results (see below), we propose the following conclusions. Comments welcome.

- a. EAP Notification Requests may be sent at any time by the authenticator. For example, they may arrive in the middle of a method exchange. An EAP Notification Request cannot be NAK'd.
- b. An EAP Notification Response MUST be sent by the peer in response to an EAP Notification Request. The Response is sent automatically by the peer and confirms receipt, but does not indicate whether the user took any action in response to the message, or even whether the message

Esteem

Informational

[Page 47]

INTERNET-DRAFT

ESTEEM

9 January 2003

was displayed to the user at all (e.g. it may be consumed by an embedded device).

- c. EAP Notification Requests SHOULD be displayed or logged by the peer. It cannot be assumed that Notification Requests are available for processing by the running method.
- d. EAP Notification Requests or Responses do not result in a state change.
- e. EAP Notification is **not** an error indication.

Below find the original survey and posted responses.

Date: Wed, 2 Oct 2002 12:22:42 -0700 (PDT)
From: Bernard Aboba <aboba@internaut.com>
To: eap@frascone.com
Subject: [eap] Notification survey

In the EAP Design team meeting today, we discussed the use of EAP Notification. The question arose as to what people are using this for today, and why. If you have implemented EAP Notification in your method (or your EAP implementation), can you let us know:

- a. Whether you are using Notification as part of the method (e.g. Notification Request is delivered to the method)
- b. Whether you allow Notification to be sent at any time (in the middle of a method exchange)

c. Whether Notification results in a state change within the EAP state machine or not

d. Whether you would be affected if EAP Notification were to be eliminated or deprecated, and why this would be bad (or good)

--__--__--

Date: Wed, 02 Oct 2002 23:02:10 +0200
To: Bernard Aboba <aboba@internaut.com>
From: Jacques Caron <Jacques.Caron@IPsector.com>
Subject: Re: [eap] Notification survey
Cc: eap@frascone.com

Hi,

I don't use it (yet), but:

a. I definitely believe a method on the client side shouldn't expect any such messages (there should be a generic method handler for it that will

Esteem

Informational

[Page 48]

INTERNET-DRAFT

ESTEEM

9 January 2003

just display the message to the user - a bit like there should be a generic identity method handler that just returns the identity).

b. It should be possible to possible to send this nearly at any time:
- there should be only one outstanding request (i.e. not possible to send a Notification request just after some other method request before the response came back)
- at the very least it should be possible to use it between two requests from different methods, not necessarily in the middle of the exchanges of one given method.

c. Receiving a Notification request should prompt a Notification response, and then return to the same state. If the message is displayed via some kind of alert/dialog box (that requires user action), the question of whether the response is sent immediately or once the user acknowledges the dialog box remains open (in that last case there would be an intermediate state, I guess).

d. I believe it's useful to be able to send clear-text messages to the

client, in the following (non-exhaustive list of) cases:

- an authentication method failed, and the method wants to give the user more information about why
- a method (maybe not an authentication one, more an authorization one) got a NAK back, and the method wants to tell something to the user anyway (e.g. connection cost)

Of course the problem of the authenticity of the message is present, but that's a more general problem of EAP messages...

An important point is that one should just not consider that the message will actually be seen and/or interpreted by anyone, it's purely informative.

Just my 0.02 EUR.

Jacques.

--__--__--

Date: Thu, 03 Oct 2002 13:00:01 +0300

From: Preeti Vinayakray <preetida.vinayakray-jani@nokia.com>

Organization: Nokia Research Center

To: ext Bernard Aboba <aboba@internaut.com>

CC: eap@frascone.com

Subject: Re: [eap] Notification survey

Hello:

Following ans. relevant to my EAP implementation

ext Bernard Aboba wrote:

Esteem

Informational

[Page 49]

INTERNET-DRAFT

ESTEEM

9 January 2003

- > In the EAP Design team meeting today, we discussed the use of EAP
- > Notification. The question arose as to what people are using this for
- > today, and why. If you have implemented EAP Notification in your method
- > (or your EAP implementation), can you let us know:
- >
- > a. Whether you are using Notification as part of the method (e.g.
- > Notification Request is delivered to the method)
- >

Yes, it is used as a part of the method

>

> b. Whether you allow Notification to be sent at any time (in the middle of
> a method exchange)
>

No,

>
> c. Whether Notification results in a state change within the EAP state
> machine or not
>

No

>
> d. Whether you would be affected if EAP Notification were to be eliminated
> or deprecated, and why this would be bad (or good)
>

This very much depends on how implementation is done. In terms of my
implementation I consider it is 'good' as a part of methods I have
supported, But it does not change any state info for client/peer. It just
provides some line display. So elimination of this not going to affect.

-Preeti

>From carlsonj@phorcys.east.sun.com Fri Oct 4 09:53:16 2002
To: Bernard Aboba <aboba@internaut.com>
Subject: Re: [eap] Notification survey

I have this implementation:

<ftp://playground.sun.com/pub/eap/index.html>

and I'm working on updating it and integrating it into 'pppd' (since
I'm now one of the maintainers of that software) for the next release.

Esteem

Informational

[Page 50]

INTERNET-DRAFT

ESTEEM

9 January 2003

> a. Whether you are using Notification as part of the method (e.g.
> Notification Request is delivered to the method)

No, it's not part of the method.

> b. Whether you allow Notification to be sent at any time (in the middle of
> a method exchange)

Sure.

> c. Whether Notification results in a state change within the EAP state
> machine or not

No. Nothing in the RFC describes it being used that way ...

> d. Whether you would be affected if EAP Notification were to be eliminated
> or deprecated, and why this would be bad (or good)

I wouldn't be affected. I never send them, and I just log them if received (and of course send ack).

I don't think it would be good to remove this message. I like having a "I think this is interesting for you to log" message for debug and deployment reasons. If, however, we fixed 'Success' and 'Failure' to be able to include a debug text message (as I think they ought to have from the start), then I'd be somewhat happier with losing Notification. (Not perfect, but good enough.)

(I also think that the possible i18n concerns and the spamming concerns are misplaced -- the receiver can do anything it likes with these, including discarding the text.)

--

James Carlson, Solaris Networking	<james.d.carlson@east.sun.com>
SUN Microsystems / 1 Network Drive	71.234W Vox +1 781 442 2084
MS UBUR02-212 / Burlington MA 01803-2757	42.497N Fax +1 781 442 1677

Date: Thu, 3 Oct 2002 09:54:36 -0700 (PDT)
From: Bernard Aboba <aboba@internaut.com>
To: eap@frascone.com
Subject: [eap] RE: Notification survey

For what it's worth, here are some details on how Windows implements Notification:

- EAP-Notification is out-of-band. The EAP layer automatically sends a Notification Response; the Notification Request is not passed to the currently running method.

- EAP-Notification can be sent at any time. There are no checks done when the EAP-Notification arrives.
- Since EAP-Notification is not passed to the method, it does not result in a state change.
- EAP Notification does not have support for internationalization. The lack of language support means that there is no easy way to display a message to the user in a language that they can be guaranteed to understand.
- EAP-Notification does not result in a display to the user at present.

--__--__--

5. Normative references

- [RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, [RFC 1661](#), July 1994.
- [RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", [RFC 1994](#), August 1996.
- [RFC2044] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", [RFC 2044](#), October 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.]
- [RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", [RFC 2279](#), January 1998.
- [RFC2434] Alvestrand, H. and Narten, T., "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 2434](#), October 1998.
- [RFC2988] Paxson, V., Allman, M., "Computing TCP's Retransmission Timer", [RFC 2988](#), November 2000.
- [IEEE802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, ANSI/IEEE Std 802, 1990.
- [IEEE8021X] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std 802.1X-2001, June 2001.

INTERNET-DRAFT

ESTEEM

9 January 2003

6. Informative references

- [DECEPTION] Slatalla, M., and Quittner, J., "Masters of Deception." HarperCollins, New York, 1995.
- [RFC1510] Kohl, J., Neuman, C., "The Kerberos Network Authentication Service (V5)", [RFC 1510](#), September 1993.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", [RFC 2246](#), November 1998.
- [RFC2486] Beadles, M., Aboba, B., "The Network Access Identifier", [RFC 2486](#), January 1999.
- [RFC2401] Atkinson, R., Kent, S., "Security Architecture for the Internet Protocol", [RFC 2401](#), November 1998.
- [RFC2408] Maughan, D., Schertler, M., Schneider, M., Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", [RFC 2408](#), November 1998.
- [RFC2433] Zorn, G., Cobb, S., "Microsoft PPP CHAP Extensions", [RFC 2433](#), October 1998.
- [RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and Palter, B., "Layer Two Tunneling Protocol L2TP", [RFC 2661](#), August 1999.
- [RFC2716] Aboba, B., Simon, D., "PPP EAP TLS Authentication Protocol", [RFC 2716](#), October 1999.
- [KRBATTACK] Wu, T., "A Real-World Analysis of Kerberos Password Security", Stanford University Computer Science Department, <http://theory.stanford.edu/~tjw/krbpass.html>
- [KRBLIM] Bellovin, S.M., Merritt, M., "Limitations of the kerberos authentication system", Proceedings of the 1991 Winter USENIX Conference, pp. 253-267, 1991.
- [KERB4WEAK] Dole, B., Lodin, S., and Spafford, E., "Misplaced trust: Kerberos 4 session keys", Proceedings of the Internet Society Network and Distributed System Security

Symposium, pp. 60-70, March 1997.

[PIC] Sheffer, Y., Krawczyk, H., Aboba, B., "PIC, A Pre-IKE Credential Provisioning Protocol", Internet draft (work in progress), [draft-ietf-ipsra-pic-05.txt](#), February 2002.

Esteem

Informational

[Page 53]

INTERNET-DRAFT

ESTEEM

9 January 2003

[PPTPv1] Schneier, B, Mudge, "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol", Proceedings of the 5th ACM Conference on Communications and Computer Security, ACM Press, November 1998.

[IEEE8023] ISO/IEC 8802-3 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, (also ANSI/IEEE Std 802.3-1996), 1996.

[IEEE80211] Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1997, 1997.

Acknowledgments

Many thanks to the volunteers of the EAP Design team.

Authors' Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Email: bernarda@microsoft.com
Phone: +1 425 706 6605

Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.
This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into

Esteem

Informational

[Page 54]

INTERNET-DRAFT

ESTEEM

9 January 2003

languages other than English. The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

INTERNET-DRAFT

ESTEEM

9 January 2003

Open issues

Open issues relating to this specification are tracked on the following web site:

<http://www.drizzle.com/~aboba/EAP/eapissues.html>

Expiration Date

This memo is filed as <[draft-ietf-eap-esteem-01.txt](#)>, and expires July 24, 2003.

