IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resolution of Final Errata on SAE | | | | |
| Date: 2024-05-06 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE |  |  |  |
| Mario Lilli | University of Milan, Italy |  |  | mario.lilli@unimi.it |
| Roberto Metere | University of York, UK |  |  | roberto.metere@york.ac.uk |
| Luca Arnaboldi | University of Birmingham, UK |  |  | l.arnaboldi@bham.ac.uk |
| Elvinia Riccobene | University of Milan, Italy |  |  | elvinia.riccobene@unimi.it |

Abstract

A set of errata on the SAE protocol were sent to the 802.11 chair. Most of these errata were addressed in 11-24/0027r3. This document addresses the rest.

*First Erratum [Password identifier]*

*Location:* §12.4.5.4, pg. 2463.

*Original Text: If the peer’s SAE Commit message contains a password identifier, the value of that identifier shall be used in construction of the password element (PWE) for this exchange*

*Correction:* Complete with further instructions, considering Mesh connections (which is included as part of the standard: ”SAE shall be implemented on all mesh STAs to facilitate and promote interoperability.“) and what behaviour should be followed by participants that both send a *different* password identifier to their peer.

*Rationale:* It is not clear what exact operations the receiver is supposed to perform (as this is underspecified in the standard); but we guess that it should at least change EA (and made this assumption when modelling this), however this should be made explicit. In particular, the scenario of what happens if each party sends a *different* password identifier to their peer is left unexplained. We modelled this scenario where both follow the current specs (interpreted as updating EA) with ProVerif (a state-of-the-art symbolic verification tool) and can mimic a deadlock situation, results show this underspecification may lead to a DoS attack.

Discussion: There is a 1:1 binding between a password identifier and a password. While discrete passwords can have the same value and be identified by different identifiers they are treated as different (even though the value of the password is identical). There is only 1 password used in a single SAE exchange and therefore the same identifier will identify it on both sides. It is possible that a STA could have multiple passwords + identifiers but in a non-infrastructure BSS (e.g. mesh) this doesn’t make sense, so best to just mention that.

Proposed Resolution: instruct the editor to modify section 12.4.3 as indicated:

**12.4.3 Representation of a Password**

Similarly, to address ambiguity when identifying passwords, a STA shall represent a password identifier as a UTF-8 string that is processed according to the UsernameCasePreserved profile of IETF RFC 8265, the output of which is an octet string that is stored in the dot11RSNAConfigPasswordValueTable. When a “password identifier” is called for in the description of SAE that follows, the identifier from the dot11RSNAConfigPasswordValueTable is used. A mesh STA shall not use a password identifier.

*Eigth Erratum [Conflation of Participants]*

*1) Location:* §12.4.8.3.1, pg. 2469;

*2) Original Text: The SME signals the following events to the parent SAE process: — Initiate. An Initiate event is used to instantiate a protocol instance to begin SAE with a designated peer. — Kill. A Kill event is used to remove a protocol instance with a designated peer.*

*3) Correction:* A section should be created for the SME participant, to clarify its role in the protocol, just like for the other participants.

*4) Rationale:* The SME is a participant that is part of the protocol, just like the Parent process and protocol instances, so it is important to separate it from the Parent process logic. This will make it easier to add new events or outputs to the SME, as well as make the specification more semantically correct.

Discussion: The SME exists outside of SAE while the Parent process and Protocol Instance are defined by SAE. It is not part of the protocol. Therefore what is being described in the SAE state machine is how an SAE-defined entity (the Parent Process) interacts with the SME, it is not intended to be a description of the SME which exists outside of the 802.11 standard.

Proposed Resolution: rejected

*Nineteenth Erratum [Missing Del event, status code or silent deletion]*

*1) Location:* §12.4.8.6.4, pg. 2473;

*2) Original Text: The mesh STA, with the numerically lesser of the two MAC addresses, zeros Sync, shall increment Sc, choose the group from the received SAE Commit message, generate new PWE and new secret values according to 12.4.5.2, process the received SAE Commit message according to 12.4.5.4, generate a new SAE Commit message and SAE Confirm message, and shall transmit the new Commit and Confirm to the peer. It shall then transition to Confirmed.*

and

*If the received element and scalar differ from the element and scalar offered, the received SAE Commit message shall be processed according to 12.4.5.4, the Sc counter shall be incremented (thereby setting its value to one), the protocol instance shall then construct an SAE Confirm message, transmit.*

*3) Correction:* After the following sentence: ’SAE Commit message according to 12.4.5.4’ should be added: ’the protocol instance shall send a Del event to the parent process and silently discard the message;’

*4) Rationale:* As specified in the previous errata, it is necessary to handle all the errors that may arise; otherwise, unexpected behaviour in the state machine may occur. In these two cases, without a Del event, an implementation could mistakenly choose to continue the execution and transition to the next state or stop without providing any information about its current state to the parent process.

Discussion: If a Del event is sent when the STA with a numerically lesser MAC address receives an (otherwise) acceptable Commit that uses a different group then it will cease communication. The parent process would delete the protocol instance. That is not what we want. We want the protocol to succeed, albeit with a group that differs from the group initially selected by the STA’s protocol instance. But if processing fails then a Fail event should be sent.

Proposed Resolution: instruct editor to modify section 12.4.8.6.4 as indicated:

**12.4.8.6.4 Protocol instance behavior—Committed state**

* If the group is supported but does not match that used when the protocol instance constructed its SAE Commit message, DiffGrp shall be set and the local identity and peer identity shall be checked.
  + The mesh STA with the numerically greater of the two MAC addresses shall discard the received SAE Commit message, retransmit its last SAE Commit message, set the t0 (retransmission) timer, and remain in Committed state.
  + The mesh STA with the numerically lesser of the two MAC addresses shall set Sync to zero, increment Sc, choose the group from the received SAE Commit message, generate new PWE and new secret values according to 12.4.5.2 (PWE and secret generation), and process the received SAE Commit message according to 12.4.5.4 (Processing of a peer’s SAE Commit message). If processing fails, the protocol instance shall generate a Fail event and terminate, otherwise it shall generate a new SAE Commit message and SAE Confirm message, and transmit the new Commit and Confirm to the peer. It shall then transition to Confirmed state.
* If the group is supported and matches that used when the protocol instance constructed its SAE Commit message, the protocol instance checks the peer-commit-scalar and **PEER-COMMIT-ELEMENT** from the message. If they match those sent as part of the protocol instance’s own SAE Commit message, the frame shall be silently discarded (because it is evidence of a reflection attack) and the t0 (retransmission) timer shall be set. If the received element and scalar differ from the element and scalar offered, the received SAE Commit message shall be processed according to 12.4.5.4 (Processing of a peer’s SAE Commit message). If processing fails, the protocol instance shall generate a Fail event and terminate, otherwise Sc shall be incremented (thereby setting its value to one), the protocol instance shall then construct an SAE Confirm message, transmit it to the peer, and set the t0 (retransmission) timer. It shall then transition to Confirmed state.

*Twenty-Second Erratum [PT = 1]*

*1) Location*: §12.4.4.3.3, pg. 2461;

*2) Original Text*: The probability of PT taking the value 1 is to be neglected

*3) Correction*: Change it to “In the negligible probability of PT taking the value 1, a constant string is concatenated to the password, recursively until required”.

And modify the related algorithm to be in a loop

Discussion: with the defined groups, PT is never going to be 1 but algorithmically it could so it makes sense to come up with a backwards-compatible way to ensure it never does.

Proposed Resolution: instruct the editor to modify section 12.4.4.3.3 as indicated:

**12.4.4.3.3 Direct generation of the password element with FFC groups**

To perform the hash-to-element method, HKDF (IETF RFC 5869) is passed a salt in the form of the SSID for which the password is to be used, the password, optionally a password identifier, as an input key, a constant label “SAE Hash to Element”, and the length of the prime to produce a password value. The resulting password value shall be reduced into a range such that 1 < pwd-value < p. Then, it shall be raised to the power (p-1) / q and reduced modulo p (where p is the prime number and q is the order). This ensures PT is a generator of order either 1 (if PT = 1) or q (for all other values). The probability of PT taking the value 1 is negligible, but if it happens the label is appended with a constant message and PT is generated again.

This secret PT is stored until needed to generate a session specific PWE.

Algorithmically, this process is as follows:

*label = “SAE Hash to Element”*

*do {*

*len = olen(p) + ⎣ olen(p)/2 ⎦*

*pwd-seed = HKDF-Extract(ssid, password [|| identifier])*

*pwd-value = HKDF-Expand(pwd-seed, label, len)*

*pwd-value = (pwd-value modulo (p – 2)) + 2*

*PT = pwd-value(p-1)/q modulo p*

*label = label || “PT was one”*

*} while (PT==1)*

*This is actually a duplicate of #19*

*Twenty-Fifth Erratum**[Missing Del event, status code or silent deletion]*

*1) Location*: §12.4.8.6.4, pg. 2473;

*2) Original Text:* The mesh STA, with the numerically lesser of the two MAC addresses, zeros Sync, shall increment Sc, choose the group from the received SAE Commit message, generate new PWE and new secret values according to 12.4.5.2, process the received SAE Commit message according to 12.4.5.4, generate a new SAE Commit message and SAE Confirm message, and shall transmit the new Commit and Confirm to the peer. It shall then transition to Confirmed.

and

If the received element and scalar differ from the element and scalar offered, the received SAE Commit message shall be processed according to 12.4.5.4, the Sc counter shall be incremented (thereby setting its value to one), the protocol instance shall then construct an SAE Confirm message, transmit.

*3) Correction*: After the following sentence: ’SAE Commit message according to 12.4.5.4’ should be added: ’the protocol instance shall send a Del event to the parent process and silently discard the message;’

*4) Rationale*: As specified in the previous errata, it is necessary to handle all the errors that may arise; otherwise, unexpected behaviour    in the state machine may occur. In these two cases, without a Del event, an implementation could mistakenly choose to continue the execution and transition to the next state or stop without providing any information about its current state to the parent process.

Discussion: This is incorrect. The protocol instance shall “process the received SAE Commit message according to 12.4.5.4.” It doesn’t discard the frame and it should not send a Del event since it is transitioning into *Confirmed* state.

Proposed Resolution: This is a duplicate errata.

**References:**