IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolution for CIDs related to Protected BA Procedure |
| Date: January 14, 2021 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Abhishek Patil | Qualcomm Inc |  |  | appatil@qti.qualcomm.com |
| George Cherian |  |  |  |
| Jouni Malinen |  |  |  |
| Alfred Asterjadhi |  |  |  |
| Duncan Ho |  |  |  |
| Gaurang Naik |  |  |  |
| Yanjun Sun |  |  |  |
| Abdel Karim |  |  |  |
| Nehru Bhandaru | Broadcom |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

 Abstract

This submission proposes resolutions for comments related to protected BA feature that were received in LB258 (for REVme D1.0).

***TGm editor: The baseline for this document is REVme D1.0.***

**Revisions:**

* Rev 0: Initial version of the document.
* Rev 1: Updated based on offline feedback from various members
* Rev 2: Additional updates based on further feedback from members

Interpretation of a Motion to Adopt

A motion to approve this submission means that the editing instructions and any changed or added material are actioned in the TGm Draft. This introduction is not part of the adopted material.

Part 1: Indication of support for protected BA and inter-op issues with legacy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1002 | Abhishek Patil | 1300.00 | 64 | 9.4.2.24.4 | Based on discussion with several members (affiliated with different organizations), it was discovered that many devices (already deployed in the field) are incorrectly setting the PBAC bit in RSN Capabilities field to 1 even when they do not support protected BA operation. Future amendments are expects to mandate support for protected BA procedure (given the increasing vulnerabilities). Incorrect setting of PBAC bit will create inter-op issues when the peer device that supports protected BA procedure, expects the other side to follow the same procedure. | The commenter will provide a contribution to address this issue. | **Revised**Agree with the comment. Advertising PBAC while the device doesn’t support the procedure for protected BA agreement will lead to interop issues when the peer device supports this feature. Future amendments are expected to encourage protected BA agreements to thwart any attacks that exploit the vulnerabilities of BAR and existing BA procedure.The proposed change sets the exiting PBAC bit to reserve and defines a new bit in the RSN Extension element.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1002** |
| 1821 | Mark RISON | 2282.00 | 60 | 10.25.7 | "A STA indicates support for protected block ack by setting the RSN Capabilities field subfields MFPC, MFPRand PBAC to 1. " -- it is not clear why MFPR need be 1 (at least if dot11RSNAPBACRequired is not true) | Delete ", MFPR" from the cited text | **Revised**Agree with the comment. A PBAC STA can negotiate an unprotected BA agreement with a non-PBAC STA (see REVme D1.0 P2283L5), therefore MFPR is not a requirement to be a PBAC. A PBAC STA can establish a protected BA agreement with another PBAC STA without requiring MFPR set to 1.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1821** |

**Discussion:**

The IEEE 802.11 standard defines a mechanism to setup protected block ack (BA) session between an originator and a recipient. A device indicates support for protected BA by setting the PBAC, MFPR and MFPC subfield in the RSN Capabilities field. However, it was determined that there exist devices in the field which can incorrectly indicate support for protected BA, by setting the related bits if the peer STA has set them to 1. This issue was discovered during debugging of some issues related to RSN capabilities field – specifically when additional tests were run where the peer device was configured to indicate support for protected BA. Such ‘copying’ of PBAC capability will lead to inter-op issues when a peer device genuinely supports protected BA feature. To address this issue, we need to change the fields that are used to indicate support for protected BA. Fortunately, we are not aware of any existing implementations of protected BA. This is based on our discussion (for several months) with various members, affiliated with different 802.11 product vendors. In addition, an email was sent on January 13th, 2022, to the IEEE 802.11 reflector soliciting a response if anyone was aware of an existing implementation of protected BA. We have not received any responses until the time this document was updated (January 20th, 2022). As a result, we don’t expect any impact to existing devices, in the field, with the proposed signaling changes in this document.

------ End of discussion -----

* RSN capabilities[1002]

***TGm editor: Please update Figure 9-350 to change bit 12 (B12) from “PBAC” to “Reserved”***

***TGm editor: Please update the following bullet in this subclause as shown below:***

* Bit 12: Reserved.
* RSN Extension element (RSNXE)[1002]

***TGm editor: Please add a new row to Table 9-363 as shown below and accordingly update the content of the last row corresponding to ‘Reserved’ values:***

|  |
| --- |
| * Extended RSN Capabilities field
 |
| Bit | Information | Notes |
| <ANA> | PBAC | A STA sets the PBAC field to 1 to indicate it can establish a protected block ack agreement and sets it to 0 otherwise. |

* **Protected block ack agreement**

***TGm editor: Please update the 1st paragraph this subclause as shown below:***

[1002, 1821]A STA indicates support for protected block ack by setting the MFPC subfield in the RSN Capabilities field to 1 (see 9.4.2.24.4 (RSN Capabilities)) and the PBAC subfield in the Extended RSN Capabilities field to 1 (see 9.4.2.241 (RSN Extension element (RSNXE))). Such a STA is a PBAC STA; otherwise, the STA is a non-PBAC STA. A block ack agreement that is successfully negotiated between two PBAC STAs is a protected block ack agreement. A block ack agreement that is successfully negotiated between two STAs when either or both of the STAs is not a PBAC STA is a block ack agreement that is not a protected block ack agreement.

Part 2: Update side note in Fig 5-1 for protected BA case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1001 | Abhishek Patil | 355.00 | 45 | 5.1.5.1 | In a protected BA agreement, the order of decryption and update to reorder buffer must be maintained (as shown in Figure 5-1). This will prevent an attack scenario where a rogue device injects fake Data frame(s) which causes WinStartB to get updated while the decryption of the frame(s) fails. Update the side note shown in the figure to capture this intention. | The commenter will provide a contribution to address this. | **Revised**Agree with the comment. Changing the order of decryption/integrity check and reorder buffering will enable an attack scenario where the reorder buffer is filled with (fake) Data frames and the attacker is able to update the WinStartB value. The proposed change updates the side note in Figure 5-1 requiring that the order of the processes is maintained in protected BA agreement.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1001** |

**5.1.5 MAC data service architecture**

**5.1.5.1 General**[1001]

***TGm editor: Please replace the side note in Figure 5-1 as shown below:***



Part 3: Missing details/bugfixes to existing text in 10.25.7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1017 | Abhishek Patil | 2283.00 | 31 | 10.25.7 | WinStartB must not be updated when an error/attack condition is determined (i.e., dot11PBACErrors is incremented). | The commenter will provide a contribution to address this. | **Revised**Agree with the comment. The proposed text change states that the recipient STA does not use the value carried in Block Ack Starting Sequence Control subfield of the BAR frame to WinStartB. As a result, this statement will also cover the error condition stated in subsequent sentences.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1017** |
| 1014 | Abhishek Patil | 2283.00 | 26 | 10.25.7 | A BAR frame is not protected. Therefore, in a protected BA setup, the Block Ack Starting Sequence Control subfield value in a BAR frame must not be used for updating the WinStartR value. | The commenter will provide a contribution to address this. | **Revised**Agree with the comment. Since BAR is not a protected frame, the proposed change states that the recipient STA must not use the value carried in Block Ack Starting Sequence Control subfield to update the value of WinStartR. Instead, the WinStartR gets updated based on the SN carried in a genuine MPDU.Since clause 10.25.6.5 specifies the rules for sending a BA. Therefore, the first sentence of the first bullet is not required.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1014**  |
| 1745 | Mark RISON | 2283.00 | 24 | 10.25.7 | “The Block Ack Starting SequenceControl subfield value may be utilized for the purposes of updating the value of WinStartR.” Is not clear. This is generally the case anyway | Prepend “NOTE---" and change “may” to “could” | **Revised**The cited sentence was updated as a resolution to CID 1014. Since BAR is not a protected frame, the proposed change states that the recipient STA must not use the value carried in Block Ack Starting Sequence Control subfield to update the value of WinStartR. **TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1014** |

* **Protected block ack agreement**

***TGm editor: Please update the 4th paragraph and its 1st bullet this subclause as shown below:***

[1014]A STA that has successfully negotiated a protected block ack agreement shall obey the following rules for that agreement as a block ack recipient in addition to rules specified in 10.25.6.3 (Scoreboard context control during full-state operation) to 10.25.6.6 (Receive reordering buffer control operation):

* [1014] [1017]The STA shall not use the Block Ack Starting Sequence Control subfield value in the BlockAckReq frame for the purposes of updating the value of *WinStartB*[1014] and *WinStartR*. If the Block Ack Starting Sequence Control subfield value is greater than *WinEndB* or less than *WinStartB*, dot11PBACErrors shall be incremented by 1. If, for a block ack agreement with segmentation and reassembly, the MPDU Starting Sequence subfield value is greater than *WinEndB* or less than *WinStartB*, dot11PBACErrors shall be incremented by 1.

Part 4: Differentiating ADDBA Request for the purpose advancing the window vs ADDBA Request for updating the parameters of an existing BA agreement

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1808 | Mark RISON |  |  | 10.25.2 | It is not clear how a modification/renegotiation ADDBA Req is distinguished from a PBAC window-moving one (which does not elicit an ADDBA Rsp). 9.6.4.1 "ADDBA Request and ADDBA Response frames are used to set up or, if a STA is PBAC, to modify Block Ack operation" suggests modification is only ever done for PBAC, but if that's the case it needs to be more explicit in Clause 10/11, and also it contradicts 10.25.2's "The originator STA may send an ADDBA Request frame in order to update block ack timeout value." (note missing article too) | In 9.6.4.1 delete ", if a STA is PBAC," and change "Block Ack operation" to "block ack operation". In 10.25.2 change "to update block ack" to "to update the block ack" | **Revised**Per the current spec, the Fragment Number subfield of the Block Ack Starting Sequence Control field is always set to 0 when ADDBA Request frame is sent for setting up a new BA agreement or to modify the parameters of an existing BA agreement. The proposed change is to use the Fragment Number subfield as a differentiate between an ADDBA Request intended to update the WinStartB and an ADDBA Request for other purpose. Fragment Number subfield set to 1 indicates ADDBA Request is intended to advance the window.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1808** |
| 1013 | Abhishek Patil | 2283.00 | 15 | 10.25.7 | At a receiver, Management frames (such as ADDBA Request) are handled by processes that come much later in the sequence while an ACK response is sent in SIFS. To prevent race condition where the originator and recipient get into different states, the originator must not update the WinStartO until it receives a confirmation (ADDBA Response frame) from the recipient in response to its ADDBA Request frame sent to update the WinStartB. Clause 9.6.4.1 (P1860L51) mentions this but the normative text and details of the procedure are not described in clause 10.25.7. | The commenter will provide a contribution to address this issue. | **Revised**Based on offline discussions, it was determined that the race condition issue won’t occur. In addition, resolution to CID 1808 provides a mechanism to differentiate between an ADDBA Request meant to update the WinStartB and an ADDBA Request meant to update the parameters of the BA agreement. The proposed change clarifies that ADDBA Response is not request if the ADDBA Request was received for the purpose of updating WinStartB.**TGm editor, please implement changes as shown in** [**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1013** |

* **General**

***TGm editor: Please update the following paragraph this subclause as shown below:***

[1808]ADDBA Request and ADDBA Response frames are used to set up or to modify Block ack operation for a specific TC, TS, or GCR group address. A Block Ack Action field, in the octet immediately after the Category field, differentiates the Block Ack Action frame formats. The Block Ack Action frames are used to negotiate several parameters of a block ack agreement and the type of BlockAck frames that are used: BlockAck, NDP BlockAck and Block Acknowledgment TWT frames (see 10.25.2 (Setup and modification of the block ack parameters)). The Block Ack Action field values associated with each frame format within the Block Ack category are defined in Table 9-442 (Block Ack Action field values).

* ADDBA Request frame format

***TGm editor: Please update the following paragraph this subclause as shown below:***

The Starting Sequence Number subfield of the Block Ack Starting Sequence Control field (see Figure 9-48 (Block Ack Starting Sequence Control subfield format)) contains the sequence number of the first or next (in the case of a renegotiation of a block ack agreement) MSDU to be sent under this block ack agreement. [1808]When ADDBA Request frame is transmitted under a protected block ack agreement for the purpose of changing the value of *WinStartB* at the receiver STA, the Fragment Number subfield is set to 1; Otherwise, the Fragment Number subfield is set to 0.

* Setup and modification of the block ack parameters

***TGm editor: Please update the following paragraph this subclause as shown below:***

[1808]The originator STA may send an ADDBA Request frame in order to update the block ack timeout value. If the updated ADDBA Request frame is accepted, both STAs initialize the timer to detect block ack timeout. Even if the updated ADDBA Request frame is not accepted, the original block ack setup remains active.

* General

***TGm editor: Please update the following paragraph this subclause as shown below:***

[1808]The originator may send a BlockAckReq frame for block ack agreement that is not a protected block ack agreement or an ADDBA Request frame, with the Fragment Number subfield of the Block Ack Starting Sequence Control field set to 1, for protected block ack agreement when a QoS Data frame that was previously transmitted within an A-MPDU that had Normal Ack ack policy is discarded due to exhausted MSDU lifetime. The purpose of this BlockAckReq or ADDBA Request frame, with the Fragment Number subfield set to 1, is to shift the recipient’s WinStartB value past the hole in the sequence number space that is created by the discarded Data frame and thereby to allow the earliest possible passing of buffered frames up to the next MAC process. Under a block ack agreement with segmentation and reassembly, the BlockAckReq frame shall contain only MPDU\_SSN and the ADDBA Request frame, with the Fragment Number subfield, shall contain only MPDU\_SSN and MSDU\_SSN fields of an MPDU that has the value of the Start of MSDU subfield equal to 1.

* **Protected block ack agreement**

***TGm editor: Please update the 3rd paragraph this subclause as shown below:***

[1808]A STA that has successfully negotiated a protected block ack agreement shall obey the following rule for that agreement as a block ack originator in addition to rules specified in 10.25.6.7 (Originator’s behavior) and 10.25.6.8 (Maintaining block ack state at the originator):

* [1808]To change the value of *WinStartB* at the receiver, the STA shall transmit an ADDBA Request frame with the Fragment Number subfield of the Block Ack Starting Sequence Control field set to 1 and the Starting Sequence Number subfield of the Block Ack Starting Sequence Control field set to the intended value.

***TGm editor: Please update the 2nd bullet of the 4th paragraph this subclause as shown below:***

* [1808]Upon receipt of an ADDBA Request frame, with the Fragment Number subfield of the Block Ack Starting Sequence Control field set to 1, for an established protected block ack agreement whose TID and transmitter address are the same as those of the block ack agreement, the STA shall update its *WinStartR* and *WinStartB* values based on the Starting Sequence Number subfield in the ADDBA Request frame according to the procedures outlined for reception of BlockAckReq frames in 10.25.6.3 (Scoreboard context control during full-state operation), 10.25.6.4 (Scoreboard context control during partial-state operation), 10.25.6.6.1 (General), and 10.25.6.6.3 (Operation for each received BlockAckReq), while treating the starting sequence number as though it were the *SSN* of a received BlockAckReq frame or, in case of a block ack agreement with segmentation and reassembly, treating the MPDU starting sequence number as though it were the MPDU SSN of a received BlockAckReq frame. The STA shall ignore the values carried in the other fields of the Action field of an ADDBA Request frame with the Fragment Number subfield set to 1. [1013]The STA shall not transmit an ADDBA Response frame in response to receiving an ADDBA Request frame with the Fragment Number subfield set to 1.

Part 5: Addressing new vulnerabilities

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CID** | **Commenter** | **Page** | **Line** | **Clause** | **Comment** | **Proposed Change** | **Resolution** |
| 1015 | Abhishek Patil | 2283.00 | 31 | 10.25.7 | In addition to the stated conditions (P2283L22), the scoreboard context at the recipient STA must not be updated if MPDU decryption or integrity check fails. This can happen when an attacker injects a fake Data frame in such case, the decryption will fail). | The commenter will provide a contribution to address this. | **Revised**Agree with the comment. The proposed change states that the recipient STA flushes the entire scoreboard context if it maintains partial state and the MPDU decryption or integrity check fails, or the recipient STA flushes the entry for the MPDU from the scoreboard context if it maintains full state and the decryption or integrity check for that MPDU fails.[**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1015** |
| 1016 | Abhishek Patil | 2283.00 | 31 | 10.25.7 | In addition to the stated conditions (P2283L22), the scoreboard context and the reorder buffer at the recipient STA must not be updated if MPDU fails replay check. This can happen when an attacker replays a valid MPDU with an updated SN. Such MPDU will pass decryption but will fail replay check. | The commenter will provide a contribution to address this. | **Revised**Agree with the comment. The proposed change states that the recipient STA flushes the entire scoreboard context and flush the entry for the MPDU from reorder buffer if it maintains partial state and the PN-based replay check failed for an MPDU that passed decryption and integrity check, or the recipient STA flushes the entry for the MPDU from the scoreboard context and the reorder buffer if it maintains full state and the PN-based replay check fails for an MPDU that passed decryption and integrity check.[**https://mentor.ieee.org/802.11/dcn/22/11-22-0082-02-000m-lb258-resolution-for-cids-related-to-protected-ba.docx**](https://mentor.ieee.org/802.11/dcn/22/11-22-0082-01-000m-lb258-resolution-for-cids-related-to-protected-ba.docx) **tagged as 1016** |

**Discussion**

The comments bring attention to two vulnerabilities:

1. Injection of a fake Data frame: Attacker injects a Data frame with an arbitrary SN. The attack won’t be detected until decryption/integrity check. But by then, the scoreboard context (and possibly *WinStartR*) will get updated.
2. Replay a genuine Data frame with a modified SN: Attacker records a genuine Data frame and replays it with a modified SN. Since it is a replayed frame, it will pass decryption and integrity check. The attack goes unnoticed until PN-based replay check is performed (which comes later in the processing chain). By the time replay check fails, the scoreboard context (and possibly *WinStartR*) will get updated. Also makes a fake entry in the reorder buffer (and possibly updates *WinStartB*).

Proposed resolutions:

1. If an MPDU does not pass decryption or integrity check and
* If the recipient STA maintains full state, then the recipient STA shall not update the value of *WinStartR* and shall clear the BA scoreboard context for that MPDU.
* If the recipient STA maintains partial state, then the recipient STA shall clear the BA scoreboard context for that MPDU.
1. If replay check fails for an MPDU that was successfully decrypted and passed integrity check and
* If the recipient STA maintains full state, then the recipient STA shall not update the value of WinStartB and WinStartR, shall clear the scoreboard context for that MPDU, and shall clear the entry for that MPDU from the reorder buffer.
* If the recipient STA maintains partial state, then the recipient STA shall not update the value of WinStartB, shall clear the scoreboard context, and shall clear the entry for that MPDU from the reorder buffer.

------ End of discussion -----

* **Protected block ack agreement**

***TGm editor: Please add a new bullets to the 4th paragraph this subclause as shown below:***

A STA that has successfully negotiated a protected block ack agreement shall obey the following rules as a block ack recipient in addition to rules specified in 10.25.6.3 (Scoreboard context control during full-state operation) to 10.25.6.6 (Receive reordering buffer control operation):

* [1015]If the STA uses full state operation and an MPDU does not pass decryption or integrity check, then the STA shall not update the value of *WinStartR*, shall clear the scoreboard context for that MPDU, and shall increment dot11PBACErrors by 1. If the STA uses partial state operation and an MPDU does not pass decryption or integrity check, then the STA shall clear the entire scoreboard context, and shall increment dot11PBACErrors by 1.
* [1016]If the STA uses full state operation and the replay check fails for an MPDU that was successfully decrypted and passed integrity check, then the STA shall not update the value of *WinStartB* and *WinStartR*, shall clear the scoreboard context for that MPDU, shall clear the entry for that MPDU from the reorder buffer, and shall increment dot11PBACErrors by 1. If the STA uses partial state operation and the replay check fails for an MPDU that was successfully decrypted and passed integrity check, then the STA shall clear the entire scoreboard context, shall not update the value of *WinStartB*, shall clear the entry for that MPDU from the reorder buffer, and shall increment dot11PBACErrors by 1.

NOTE - An implementation can buffer each received MPDU and later discarded it if replay check fails for that MPDU. Alternatively, an implementation can perform replay check before it buffers a received MPDU.