Security analysis of NWDAF analytics subscription lifecycle

This report in another language: 中文.

19.8.0 · ProVerif+ Agent report

Documents	5G System; Network Data Analytics Services; Stage 3 (protocol definition)
Properties examined	12: 4 hold, 3 fail (3 attacks), 5 without an answer
Findings	1

Contents

1 Protocol and scope
2 Findings
3 Properties examined
4 Limits of this analysis
Appendix A: Reading of each property
Appendix C: Claims and prover answers
Appendix D: How the analysis was run

1 Protocol and scope

The protocol is the Nnwdaf_EventsSubscription API of TS 29.520: an NF service consumer asks an NWDAF for network performance analytics and receives notifications at a callback URI it supplies. The parties are the consumer, the NWDAF that produces the analytics, and the NRF as the OAuth2 authorization server that issues access tokens for the API.

A run is one subscription: the consumer obtains a token, posts a subscription with its notificationURI and event, the NWDAF creates the resource and returns the assigned subscriptionId, sends up to two notifications to that URI, and later removes the subscription when the consumer sends DELETE. Each party ends holding the subscriptionId of that one resource, and only that consumer can read the analytics in the notifications.

The parties:

- **NF service consumer**: the analytics consumer; holds client credentials. Holds: client credentials, event to subscribe, notificationURI.
- **NWDAF**: produces the analytics; holds the subscription store. Holds: none beyond its identity.
- **NRF**: authorization server. Holds: registered NF identities.

Figure 1 shows the exchange as the documents specify it.

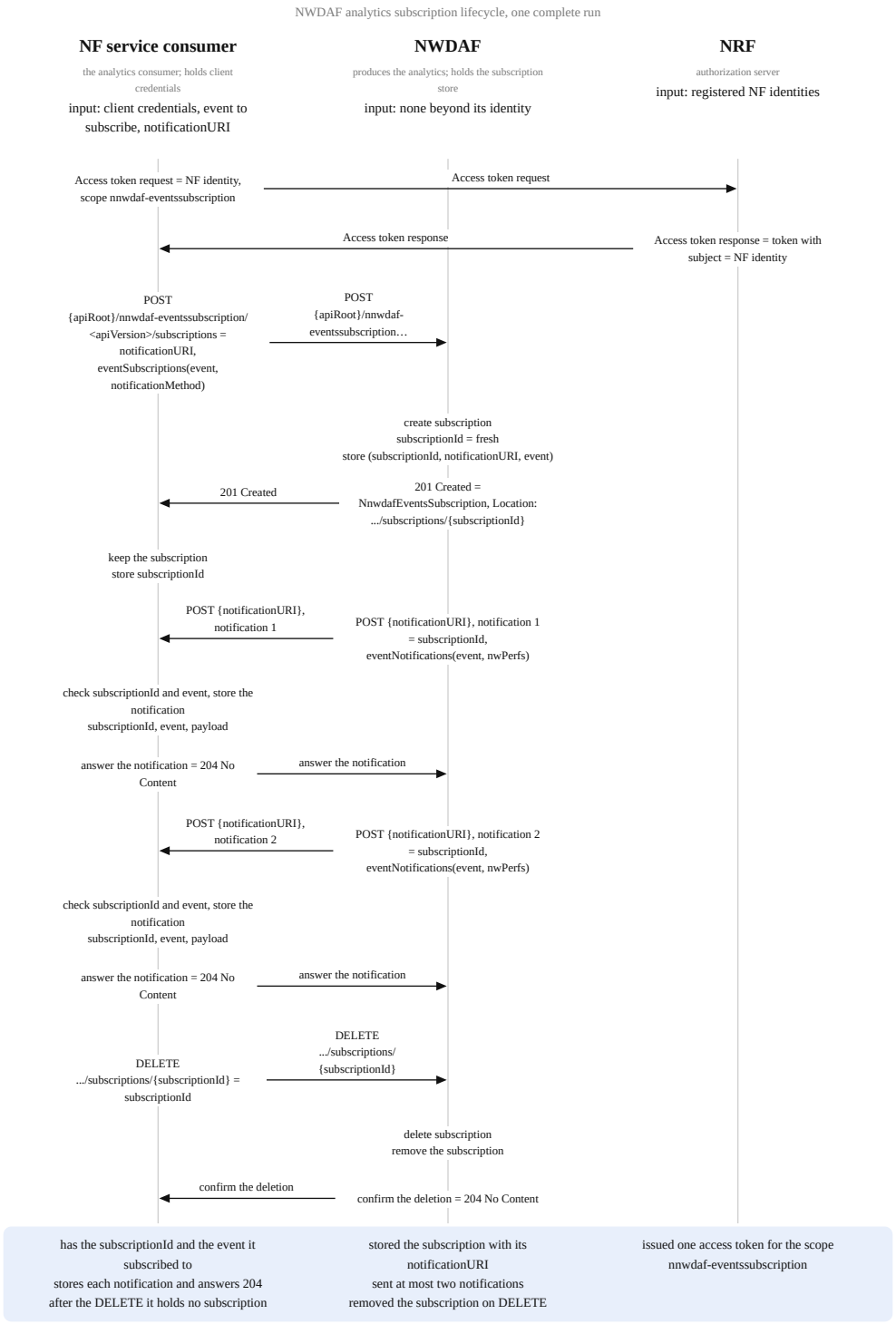


Figure 1. The protocol as specified.

The scope of the analysis is in Appendix D: what was analysed, what was left out, and the sections of the documents it is built from.

2 Findings

The analysis has 1 finding: what follows is not stated in the documents. For 1 of them it is unclear whether the documents state it.

The exchange holds against the network: no operation is accepted without a token the NRF issued, and notifications are authenticated, bound to the subscribing consumer and confidential. The analysis found one weakness the documents do not state: the NWDAF never checks that the consumer updating or deleting a subscription is the one that created it, so another NF with a valid token for the same scope can cancel someone else's subscription if it holds the subscriptionId. The checked remedy is to record the token subject with the resource and refuse an Update or an Unsubscribe from any other consumer. Five of the twelve targets, all about the lifecycle after a change or a cancellation, are answered only by reachability, because deletion and replacement cannot be represented with ProVerif's monotonic state; the other seven carry a proof or an attack, each with its control.

The attack figures use the columns of Figure 1. Red is what the attacker does. Yellow marks a message that a party sends although the specified exchange does not have it at that point, and the outcome that the attack breaks.

2.1 Another NF can unsubscribe a subscription it did not create

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (ts-33-501:2875)
Value	medium
Properties	Z03, Z03.10, A03 (section 3)
Source text	s0278, s0280, s0324, s0325
Shown by	ProVerif attack queries `own_del`, `del_a`; Figures 2, 3 and 4

What happens. The NWDAF accepts an HTTP DELETE on an Individual NWDAF Event Subscription from any NF that presents a valid token for the API, and removes the subscription named in the request. Nothing in the source text requires the caller to be the consumer that created the resource, and the model has no such check either. The subscriber loses the subscription without ever having asked to cancel it.

What the attacker needs. A legitimate NF with its own client credentials and an access token for the nnwdaf-eventssubscription scope, and knowledge of the victim's subscriptionId. The source neither binds the identifier to its creator nor requires it to be unpredictable.

What the attacker gains. Analytics delivery to the victim stops. With the same gap an Update can point the notificationURI at the attacker, which would also hand it the analytics of the victim's subscription.

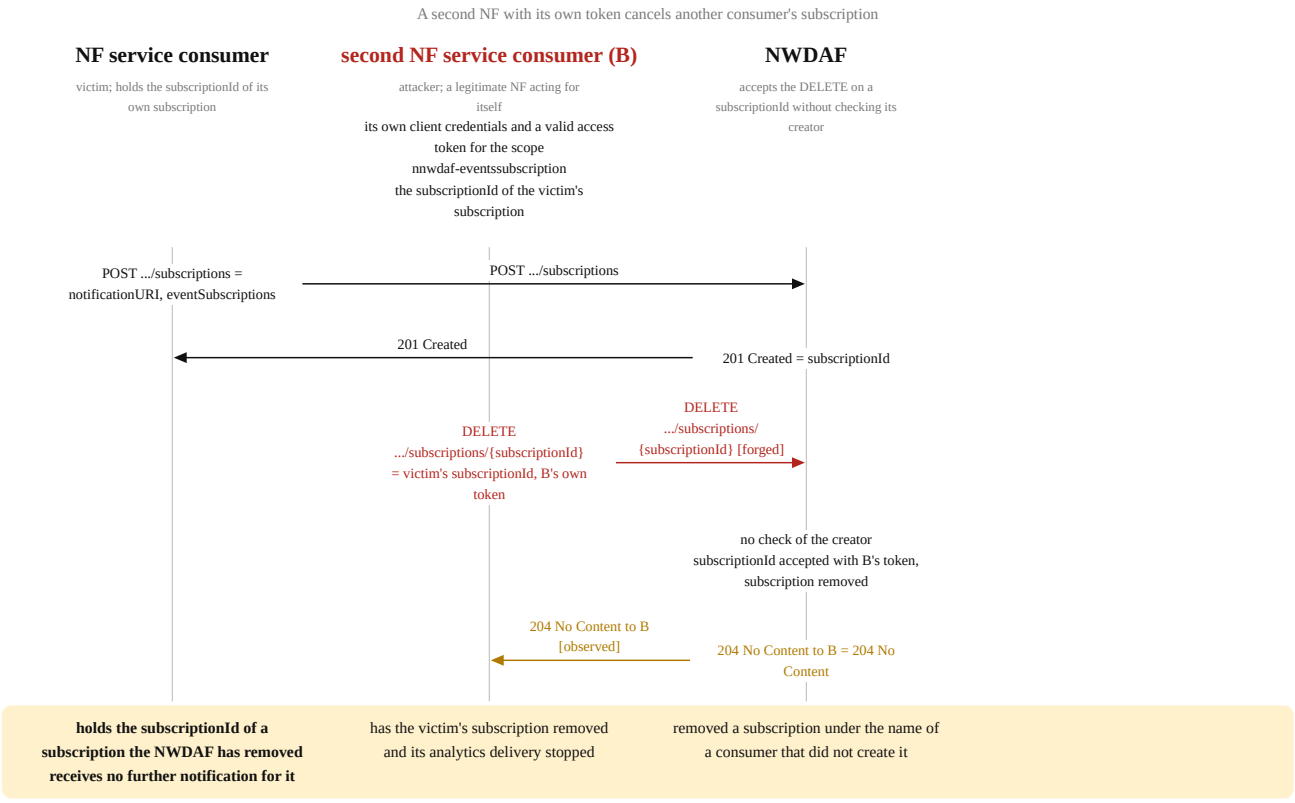


Figure 2. A second NF with its own token cancels another consumer's subscription. Property Z03; attack query own_del.

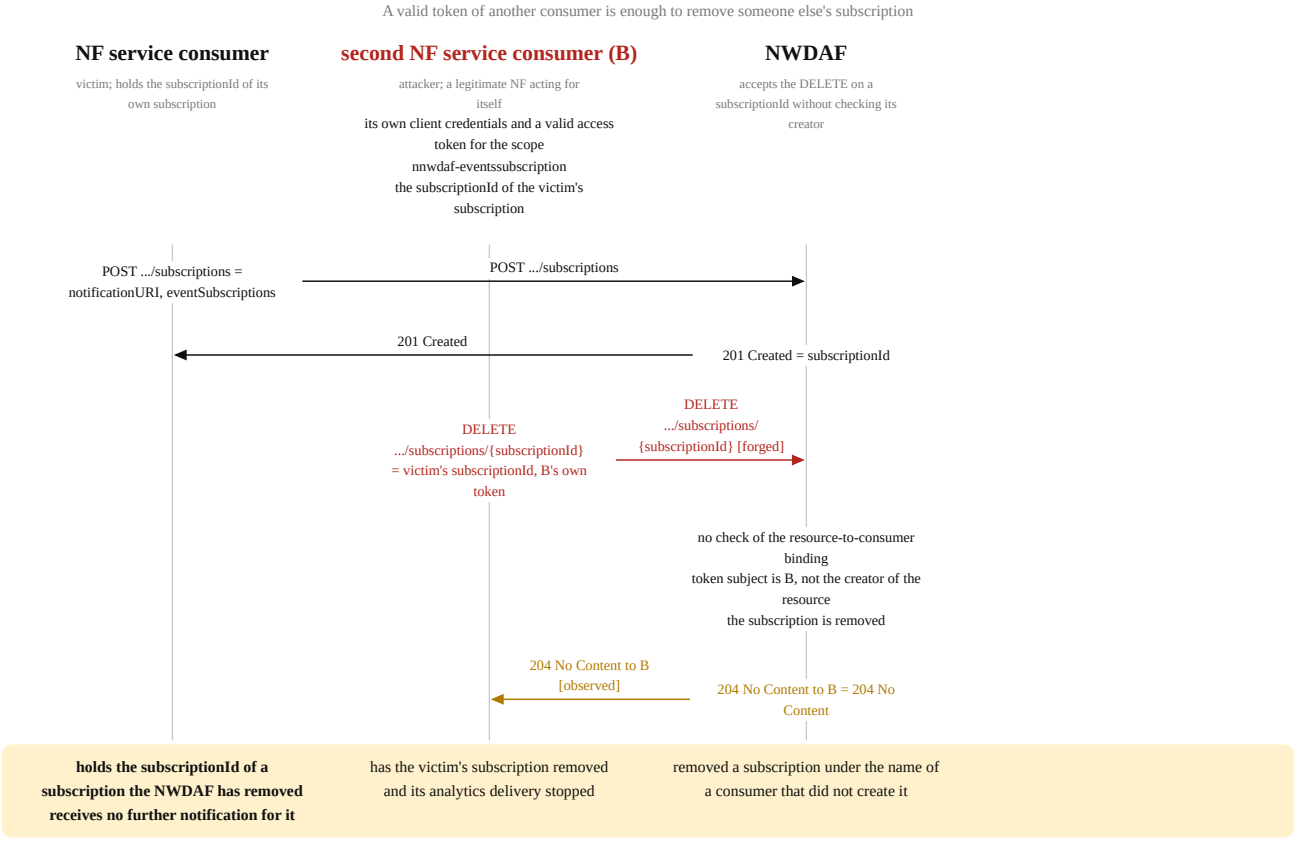


Figure 3. A valid token of another consumer is enough to remove someone else's subscription. Property Z03.10; attack query own_del.

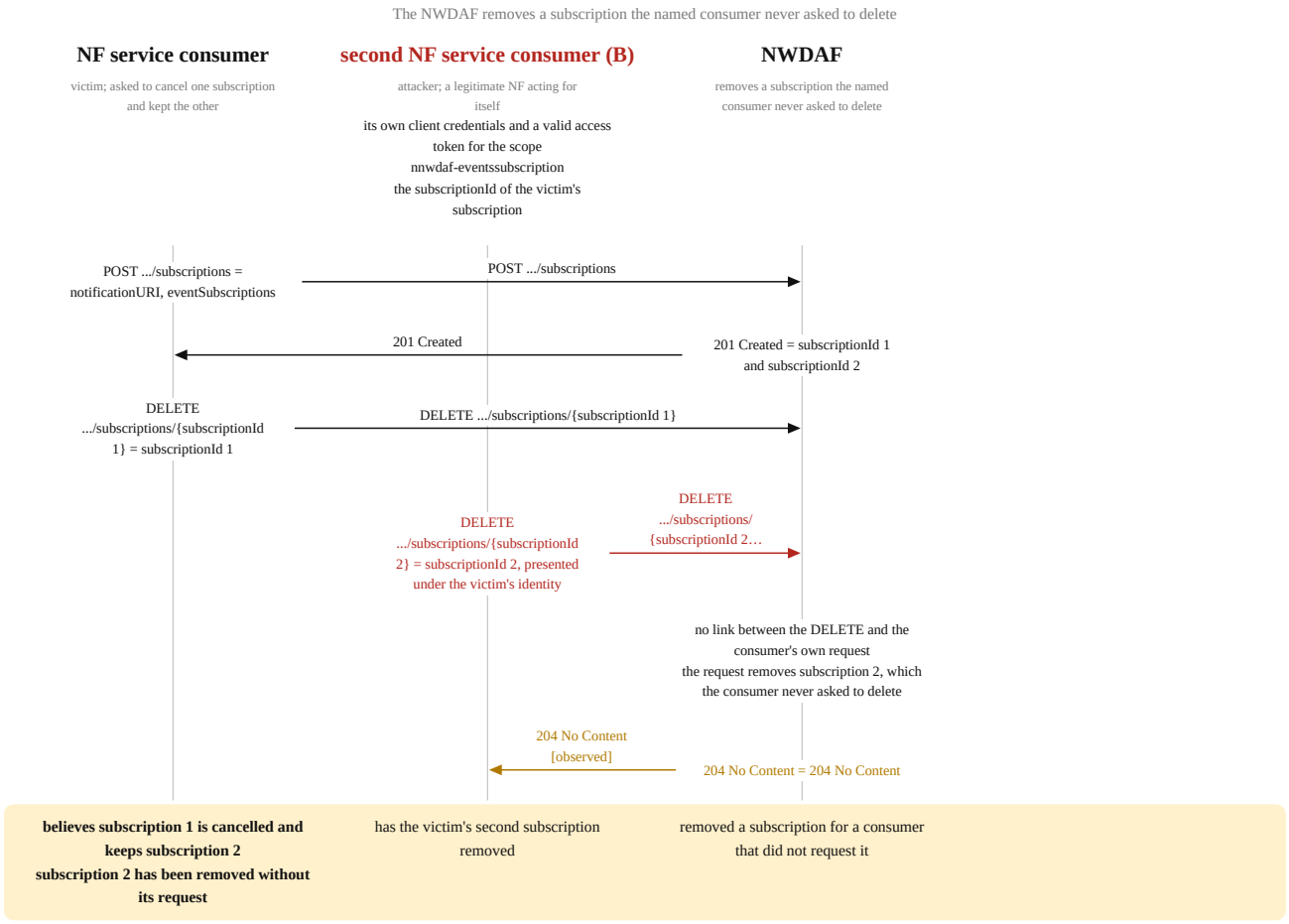


Figure 4. The NWDAF removes a subscription the named consumer never asked to delete. Property A03; attack query del_a.

Proposed change. Record the subject of the OAuth2 token with the subscription at creation, and on Update and Unsubscribe reject the request unless the token subject is that recorded consumer. The change was checked on a copy of the model: with it, the recorded attack that lets another NF delete the subscription is no longer found, the normal run is still found, and none of the other recorded answers changed.

Check of the change with ProVerif. The change was put into a copy of the model: Record the consumer that created the subscription (the subject of the OAuth2 token) with the resource, and on Unsubscribe (and, in the same way, on Update) reject the request unless the token subject is that recorded consumer. On the changed model ProVerif proves `own_del`: the attack is no longer possible there. The normal run of the protocol is still found. The other 13 recorded answers are unchanged. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0055-fix-binding.pv`.

No figure was given for these attacks of a finding: A03: own_del, Z03.10: del_a, Z03: del_a.

3 Properties examined

Each row is one property put to ProVerif. The result is ProVerif's answer on the analyst's model of the protocol; value and whether the documents already state the result are the analyst's judgement, with the reasons in Appendix A.

ID	Property	Compromise assumed	Result	Value	In the documents
Z01	**OAuth2 authorization of every operation.** Every protected operation the NWDAF performs (create, update, unsubscribe, or sending a notification to the consumer) must carry an OAuth2 token issued by the authorization server (NRF) that is valid for that NF consumer and the Nnwdaf_EventsSubscription API; when local policy requires user consent, a target UE that has not given consent must not be included in the analytics collection.	no compromise	Holds	low	yes
Z03	**Subscription bound to its consumer.** The authorized subject, audience (NWDAF instance) and scope of the OAuth2 token must match the operation being performed; the subscription's subscriptionId and notificationURI must be bound to the NF consumer that created the subscription, notifications may only be sent to the notificationURI currently bound to that subscription, and rights or notifications must not move to the wrong consumer or operation through token or subscription transfer.	no compromise	Fails (1 attack)	medium	unclear
D01	**Notification accepted only from its NWDAF.** Every event notification an NF consumer accepts corresponds to a notification genuinely sent by the NWDAF it subscribed to, with the subscriptionId, event identifier and notification content consistent, and has not been replaced or tampered with by an attacker.	no compromise	Holds	low	yes
A03	**Consumer and NWDAF agree on subscription.** When an NF consumer accepts a subscription create/update response, a real run of the NWDAF exists and both sides agree on the subscriptionId, eventSubscriptions and notificationURI; when the NWDAF accepts a cancellation request, the corresponding consumer did indeed request deletion of that subscriptionId.	no compromise	Fails (1 attack)	low	unclear
A04	**Distinct subscriptionId per creation.** Each successful subscription creation corresponds to a different NWDAF run and a different subscriptionId; deletion of the same subscriptionId takes effect at most once, and a replayed create or delete request cannot produce an extra subscription instance or delete someone else's subscription.	no compromise	Reachable only	low	no
R02	**Each request and notification accepted once.** The same logical subscription request (POST/PUT/DELETE) or the same logical notification is accepted at most once at the receiver, and retransmitted messages do not cause duplicate subscriptions, duplicate updates, or duplicate storage of notifications.	no compromise	Reachable only	low	no
S05	**No notification after cancellation.** After a subscription is cancelled (DELETE succeeds) or the NWDAF terminates it with termCause, that subscriptionId produces no further notifications and accepts no further updates; an old context cannot regain the rights of the original subscription.	no compromise	Reachable only	low	no
S04	**Updated subscription used for later notifications.** Notifications after an update use the updated notificationURI and subscription parameters; mute/unmute state is inherited consistently between updates and notifications; later phases do not use a superseded old context.	no compromise	Reachable only	low	no
K02	**Analytics payload only to its subscriber.** The event notification payload (the analytics data in eventNotifications) is visible only to the NF consumer subscribed to that event and to its NWDAF; other NFs (including legitimate NFs holding their own credentials) cannot read it.	no compromise	Holds	low	yes

ID	Property	Compromise assumed	Result	Value	In the documents
Z03.10	**Token subject and subscription ownership.** The authorized subject, audience and scope of the OAuth2 token must match the operation being performed; the subscriptionId and notificationURI must be bound to the NF consumer that created the subscription, and notifications are sent only to the notificationURI currently bound to that subscription.	no compromise	Fails (1 attack)	medium	unclear
U01	**Notifications do not cross subscriptions.** The subscriptionId of every notification a consumer accepts belongs to that consumer's own subscription, and the event data in the notification was generated by that subscription's NWDAF for that subscription and does not cross into other subscriptions or consumers.	no compromise	Holds	low	yes
U02	**No stale notifications after update or cancel.** After a subscription update or cancellation takes effect, the consumer accepts no further notifications generated from the old subscription parameters; only in-flight notifications already sent before the update or cancellation took effect may be accepted.	no compromise	Reachable only	low	no

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Whether subscriptionId, notificationURI and consumer identity are bound together	Nothing in the selected text binds a subscriptionId or a notificationURI to the consumer that created it: a token of the same scope from another NF is enough to unsubscribe a subscriptionId that NF holds, and the checked remedy is to record the token subject with the resource and compare it on Update and Unsubscribe.	Z03, Z03.10, Z01
Q2: Notification correlation and cross-subscription data isolation	Notifications are correlated by the subscriptionId, which is covered by the NWDAF's signature and bound to the subscribing consumer, and the payload is readable only by that consumer, so data does not cross subscriptions; an update that repoints the notificationURI was not modelled.	U01, D01, K02
Q3: Acceptance conditions for stale notifications after an update and a bounded-state cancellation	Only reachability was established for stale notifications after an update or a cancellation; the answer needs a model that unrolls the subscription state, which the present model cannot do with monotonic tables.	U02, S05, S04

4 Limits of this analysis

A result here is a result on a model of the protocol: it says what the specified messages and checks give or fail to give, with cryptography taken as ideal, and nothing about an implementation.

What this analysis does not cover, as the analyst states it:

- TLS is idealised as per-endpoint signatures and public-key encryption with ideal primitives; no computational or quantum resistance is claimed, and no channel binding between the token and the TLS session is modelled.
- The subscription lifecycle after an update or a cancellation is not modelled: ProVerif tables only grow, so a subscription is never really removed and the muting, replacement and termCause states are absent. The five lifecycle targets are answered by reachability only.
- The update operation (PUT) is not modelled separately; its effect on the notificationURI is discussed from the source text.
- Notifications are counted by the unrolled process (two per subscription) rather than by a maxReportNbr check, so a limit on the number of reports is not analysed.

- User consent, the UDM interaction, subscription transfer, roaming, SCP or indirect communication and redirects are outside the selected flow.
- The attacker never holds a stored subscription state: a malicious NF is modelled by giving the attacker one NF's credentials, not by letting it read a subscription store.

What the host records:

- No answer for: A04, R02, S05, S04, U02.
- The answers changed after the reading was written for: Z01, U01.
- Set by the analyst and without a proof or an attack (Appendix D): Z03 under `network`; A04 under `network`; A04 under `malicious_nf`; R02 under `network`; R02 under `malicious_nf`; S05 under `network`; S05 under `malicious_nf`; S04 under `network`; S04 under `malicious_nf`; U02 under `network`; U02 under `malicious_nf`.
- Proved without a control query (Appendix C): Z01: auth_sub, K02: payload_secret. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

What stays open for each property, and what would settle it, is in Appendix A.

Appendix A: Reading of each property

The analyst's own reading of the prover's answers, not an answer of the prover: what the answers mean for the protocol, what the reading rests on, whether it can be used as it stands, and what it is worth.

Z01: OAuth2 authorization of every operation

Every protected operation the NWDAF performs (creating, updating or unsubscribing, or sending a notification to a consumer) must carry an OAuth2 token issued by the authorization server (NRF) that is valid for that NF consumer and the Nnwdaf_EventsSubscription API; when local policy requires user consent, a target UE without consent must not be included in the analytics collection.

The answers of this target changed after this reading was written.

Every Subscribe and Unsubscribe the NWDAF accepts is preceded by an access token the NRF issued to that same NF for the nnwdaf-eventssubscription scope, and a request without a token fails. A malicious NF can invoke the API, but only in its own name. The user-consent half of the target does not arise: this flow is non-UE aggregate network performance and carries no SUPI or GPSI, so the NWDAF never enters the consent check.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The requirement is the one 5.1.9 and TS 33.501 state; the analysis reproduces it mechanically and finds no way round it.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2885, s0565, s0568, s0570
- Rests on: Tokens are modelled as a triple signed by the NRF and checked by the NWDAF; expiry, audience and revocation are not modelled.; The NRF authenticates the client before issuing a token, which the source places in TS 29.510.
- Open: Token expiry, audience and revocation; Whether a token issued for one API can be replayed at another
- Next step: Model token lifetime and audience and ask the same properties.

Z03: Subscription bound to its consumer

The authorized subject, audience (NWDAF instance) and scope of the OAuth2 token must match the operation performed; a subscription's subscriptionId and notificationURI must be bound to the NF consumer that created the subscription, notifications may only be sent to the notificationURI currently bound to that subscription, and authority or notifications must not be moved to the wrong consumer or operation through token or subscription transfer.

Notifications are bound: only the consumer the subscription belongs to can read them, and each one carries the subscriptionId and the event that consumer bound. The ownership half fails: the NWDAF will remove a subscription for any NF that presents a valid token and the right subscriptionId, because the source states no rule tying the resource to its creator.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: TS 33.501 requires each NF to validate incoming messages, and the API text says only that the NWDAF removes the subscription named in the request; neither states this weakness nor the binding that would prevent it.
- Stated in the documents: unclear. Source text it rests on: s0280, s0324, s0325, s0327
- Rests on: The attack needs the subscriptionId; the source requires only that the NWDAF assign one, not that it be unguessable or known only to its owner.; Update is not modelled, so the notificationURI redirection with the

same cause is argued from the source and not shown by the prover.

- Open: Where a subscriptionId becomes known to another NF; Whether an Update by a stranger is refused
- Next step: Model Update, and model an SCP or an identifier another NF can observe, to fix the conditions under which the identifier leaks.

D01: Notification accepted only from its NWDAF

Every event notification an NF consumer accepts corresponds to a notification genuinely sent by the NWDAF it subscribed to, with the subscriptionId, event identifier and notification content consistent, and not replaced or tampered with by an attacker.

A consumer accepts a notification only if the NWDAF signed it for that consumer and for a subscription the consumer holds, with the event it subscribed to. The network attacker can neither forge nor alter a notification, and a notification built for another consumer is rejected.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: TS 33.501 requires mutual authentication and validation of incoming messages on the SBI, which is what the proof reproduces.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2871, s0340, s0383, s0388
- Rests on: Ideal signatures and ideal public-key encryption stand in for mutual TLS; the certificate binding of the NWDAF identity to the name it is reached by is not modelled.; The consumer correlates only by subscriptionId and event; the optional notifCorrId is not modelled.
- Open: Replay of a notification inside a live subscription; Whether a consumer accepts a notification after it cancelled
- Next step: Ask the replay question on a model that unrolls the consumer's subscription state.

A03: Consumer and NWDAF agree on subscription

When an NF consumer accepts a subscription creation/update response, a real run of the NWDAF exists and both sides agree on the subscriptionId, eventSubscriptions and notificationURI; when the NWDAF accepts a cancellation request, the corresponding consumer did indeed request deletion of that subscriptionId.

The consumer's 201 Created is authentic: the subscriptionId it stores is one the NWDAF created in a real run, and a response built by the network attacker is rejected. The cancellation half does not hold in the same way: the NWDAF may accept a cancellation attributed to a consumer for a subscriptionId that consumer never asked to delete, so the two sides do not always agree on what was removed.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value low: The creation agreement follows the source; the cancellation mismatch is the same gap the Z03 finding describes, and the source does not state it.
- Stated in the documents: unclear. Source text it rests on: s0278, s0282, s0324, s0327
- Rests on: The response is bound to the requesting consumer by an identity inside the signed payload, which stands in for the TLS session.; Subscription state is not removed in the model, so agreement after several changes is not analysed.
- Open: Agreement after an update; Agreement when a DELETE is retransmitted
- Next step: Unroll creation, update and cancellation with the subscription state kept in a chain of processes.

A04: Distinct subscriptionId per creation

Each successful subscription creation corresponds to a distinct NWDAF run and a distinct subscriptionId; a deletion of the same subscriptionId takes effect at most once, and a replayed create or delete request cannot produce an additional subscription instance or delete another party's subscription.

A creation run with its own subscriptionId is reachable, which is all the recorded answer shows. Whether a replayed create adds a second instance, and whether a repeated delete can remove a second subscription, was not decided: the model has no way to count instances or to remove a subscription.

- Result: Reachable only; answers what is asked: partly; usable with caveats
- Value low: The source states no idempotency rule, so nothing is stated to reproduce; the search found no statement of a weakness either.
- Stated in the documents: no. Source text it rests on: s0280, s0325
- Rests on: Only reachability was run; no property about the number of instances or about replay holds a recorded answer.; The subscriptionId is a fresh name, so distinctness of identifiers is built into the model rather than checked.
- Open: Whether a retransmitted POST creates a second subscription
- Next step: Unroll two creations and two deletions and ask an injectivity property.

R02: Each request and notification accepted once

The same logical subscription request (POST/PUT/DELETE), or the same logical notification, is accepted at most once at the receiving end; retransmitted messages do not cause duplicate subscriptions, duplicate updates, or duplicate storage of notifications.

A notification is produced, delivered and accepted, so the run the target speaks about exists. Whether a given request or notification is accepted twice was not decided, because duplicates need a state that ProVerif cannot keep across an unbounded number of sessions.

- Result: Reachable only; answers what is asked: partly; usable with caveats
- Value low: The source gives no transaction identifier and says nothing about duplicates; no statement was found to reproduce.
- Stated in the documents: no. Source text it rests on: s0282, s0388
- Rests on: Only reachability was run.; HTTP retransmission happens below the modelled SBI, so the model has no duplicate to suppress.
- Open: Effect of a duplicated notification on the consumer's store
- Next step: Unroll the consumer to two acceptance steps and ask whether the same notification is stored twice.

S05: No notification after cancellation

After a subscription is cancelled (a successful DELETE) or the NWDAF terminates it with a termCause, that subscriptionId no longer produces new notifications and no longer accepts updates; the old context cannot regain the rights of the original subscription.

A notification is sent for a subscription that is later cancelled, so the situation is reached. Whether a notification can still be produced or accepted after the cancellation took effect was not decided, because the model cannot remove a subscription and therefore cannot represent the instant of removal.

- Result: Reachable only; answers what is asked: partly; usable with caveats

- Value low: The source states the removal (s0325..s0327) and the termCause indication (s0386) but says nothing about notifications in flight.
- Stated in the documents: no. Source text it rests on: s0325, s0327, s0386
- Rests on: Only reachability was run.; The NWDAF suppresses a notification when it has already recorded the cancellation, but the model's record only grows, so this is close to a definition rather than a checked property.
- Open: A notification produced before the DELETE and delivered after it; A termCause termination instead of a DELETE
- Next step: Unroll the NWDAF to a chain that records the cancellation and ask whether a third notification can follow it.

S04: Updated subscription used for later notifications

Notifications after an update use the updated notificationURI and subscription parameters; the mute/resume state is inherited consistently between the update and the notifications; later stages do not use the replaced old context.

A subscription is created and a notification for it is accepted, so the run exists. Whether later notifications follow the updated parameters, and whether a replaced notificationURI stops receiving, was not decided because update and replacement are not representable in the model's state.

- Result: Reachable only; answers what is asked: partly; usable with caveats
- Value low: The source states that the subscription is updated and stored and that the notificationURI may be modified; it says nothing about notifications in flight, so there is no stated result to reproduce.
- Stated in the documents: no. Source text it rests on: s0300, s0303, s0306
- Rests on: Only reachability was run.; The update operation is not modelled; the reading is limited to what s0300..s0303 say.
- Open: Whether a notification to the old notificationURI is still accepted after an update
- Next step: Unroll creation, update and two notifications with the URI as a state of the chain.

K02: Analytics payload only to its subscriber

The event notification payload (the analytics data in eventNotifications) is visible only to the NF consumer that subscribed to the event and to its NWDAF; other NFs (including legitimate NFs holding their own credentials) cannot read it.

The analytics in a notification can be read only by the consumer the subscription belongs to. Neither the network attacker nor another legitimate NF that holds its own credentials learns the payload, and the control shows that the payload of the NF the attacker controls is known, so the result is not vacuous.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: TS 33.501 requires confidentiality of SBI signalling, which is what the proof reproduces.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2871, s0340, s0565
- Rests on: Ideal public-key encryption stands in for TLS; the notification is encrypted to the subscribing consumer, which is the model's stand-in for the endpoint-authenticated TLS session.; The payload is a single opaque value; its content is not modelled.
- Open: Confidentiality when the notificationURI points at another NF's endpoint
- Next step: Model the callback URI as an address the NWDAF resolves, and ask whether another NF can make a subscription deliver to it.

Z03.10: Token subject and subscription ownership

The authorization subject, audience and scope of an OAuth2 token must match the operation being performed; the subscriptionId and notificationURI must be bound to the NF consumer that created the subscription, and notifications are sent only to the notificationURI currently bound to that subscription.

The token half holds: a token issued to one NF cannot be used by another to have an operation accepted, and the subject of the token is compared with the requesting identity. The ownership half fails exactly as in Z03: the subscription resource is not tied to the consumer that created it, so a valid token of the same scope is enough to act on another consumer's subscription.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: RFC 6749 notes that OAuth2 gives the resource server no method to check that a token was issued to the presenting client, and TS 33.501 asks every NF to validate incoming messages; neither states this weakness of this API nor the missing resource-to-consumer binding.
- Stated in the documents: unclear. Source text it rests on: rfc6749:2879, s0568, s0570, s0324
- Rests on: The token subject is compared with a plain identity in the request, not with a TLS client certificate; the source leaves that binding to TS 29.500.; The attack needs the subscriptionId, whose secrecy the source does not require.
- Open: Whether an audience or a per-resource scope would close the gap; Whether an Update by a stranger is refused
- Next step: Add an audience value naming the resource and test whether it alone prevents the other consumer's request.

U01: Notifications do not cross subscriptions

The subscriptionId of every notification the consumer accepts belongs to a subscription of its own, and the event data in the notification was generated for that subscription by that subscription's NWDAF, without crossing into other subscriptions or consumers.

The answers of this target changed after this reading was written.

A notification is only produced for a subscriptionId the NWDAF created for that same consumer, and the consumer accepts it only for a subscription of its own with the event it subscribed to. A notification produced for one consumer cannot be delivered under another consumer's subscriptionId.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: This is the integrity and authentication requirement of TS 33.501 applied to the notification, plus the subscriptionId correlation the source mandates.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2875, s0383, s0388
- Rests on: Ideal signatures and encryption stand in for the per-endpoint TLS session.; The consumer's store of its own subscriptions is a monotonic table, so it never forgets a cancelled subscription.
- Open: Cross-subscription delivery for a consumer that cancelled one of two subscriptions
- Next step: Unroll two subscriptions of the same consumer and ask the isolation question again.

U02: No stale notifications after update or cancel

After a subscription update or cancellation takes effect, the consumer no longer accepts notifications generated under the old subscription parameters; only in-flight notifications sent before the update or cancellation took effect may still be accepted.

A notification is accepted for a live subscription, so the run exists. Whether a notification generated under the old parameters is still accepted after an update or a cancellation took effect was not decided: the model cannot remove or replace a subscription, so the moment the parameters change is not represented.

- Result: Reachable only; answers what is asked: partly; usable with caveats
- Value low: The source gives when an update and a deletion take effect but no rule for notifications in flight, so nothing is stated to reproduce.
- Stated in the documents: no. Source text it rests on: s0300, s0325, s0386
- Rests on: Only reachability was run.; The consumer accepts against a table that only grows, so it can never refuse a notification because the subscription ended.
- Open: In-flight notification across the instant of cancellation; In-flight notification across an update of the notificationURI
- Next step: Unroll the consumer and the NWDAF to three steps with the subscription state carried in the chain.

Appendix C: Claims and prover answers

The prover's record for each target: the level reached, the query that reached it, whether a control was checked, and the attacks with the analyst's verdict on each.

target	level	bound	lemma	control	attacks	claims
Z01	proved		auth_sub	4 of 6		6 proved
Z03	proved		payload_secret	checked	own_del (depends_on_ambiguity)	1 proved, 1 reachable, 1 attack
D01	proved		auth_notif	checked		2 proved
A03	proved		sub_a	checked	del_a (depends_on_ambiguity)	1 proved, 1 attack
A04	reachable		ctl_bcreate			3 reachable
R02	reachable		reach_notif			3 reachable
S05	reachable		ctl_payload			3 reachable
S04	reachable		reach_notif			3 reachable
K02	proved		payload_secret	2 of 3		3 proved
Z03.10	proved		auth_del	checked	own_del (depends_on_ambiguity)	2 proved, 1 attack
U01	proved		auth_notif	checked	notif_own (model_artifact)	3 proved, 1 attack
U02	reachable		reach_notif			3 reachable

Claim by claim, in the analyst's wording of each claim; the answer to each is the prover's:

target	claim	attacker	answer	query	control
Z01	Every Subscribe the NWDAF accepts was preceded by an OAuth2 access token issued by the NRF to the same NF for the nnwdaf-eventssubscription scope; the attacker on the network holds no secret of a party.	`network`	proved	auth_sub	checked
Z01	Every Subscribe the NWDAF accepts was preceded by an OAuth2 access token issued to the same NF, including when the requester is a malicious NF holding its own valid token; such an NF can subscribe, but only in its own name.	`malicious_nf`	proved	auth_sub	none
Z01	User-consent claim: the selected flow is non-UE aggregate network performance and carries no SUPI or GPSI, so the consent gate of 4.2.2.2.2 is never entered; what stands in its place is that no Subscribe is accepted without a token the NRF issued to that NF.	`network`	proved	auth_sub	none
Z01	Every Subscribe the NWDAF accepts was preceded by a token issued to the same NF, also when the requester is a malicious NF with its own valid token.	`malicious_nf`	proved	auth_sub	checked
Z01	User-consent claim: the selected flow carries no SUPI or GPSI so no consent gate is entered; what stands in its place is that no Subscribe is accepted without a token issued to that NF.	`network`	proved	auth_sub	checked

target	claim	attacker	answer	query	control
Z01	Every Subscribe the NWDAF accepts was preceded by an OAuth2 access token issued by the NRF to the same NF for the nnwdaf-eventssubscription scope. Note on the attacker: the model publishes the second NF's client key, so the attacker in this file can act as one legitimate NF; the answer is therefore against an attacker that has the network plus one NF's credentials, which is the stronger of the two.	`network`	proved	auth_sub	checked
Z03	A notification for a subscription is readable only by the consumer that subscription belongs to, and carries the subscriptionId and event that consumer bound: another NF holding its own credentials cannot read or receive it.	`malicious_nf`	proved	payload_secret	checked
Z03	Only the NF service consumer that created a subscription can delete it with its own valid token.	`malicious_nf`	attack (depends_on_ambiguity)	own_del	
Z03	A subscription and a notification for it are reached, so the important/notification binding question is asked of a real run.	`network`	reachable	reach_notif	
D01	Every notification the NWDAF produces is produced for a subscription it created for that same consumer, is signed by the NWDAF and bound to that consumer, and carries the event the consumer subscribed to; the network attacker can neither forge nor alter it.	`network`	proved	auth_notif	checked
D01	Every notification is produced for a subscription created for that same consumer, also when the requester is a malicious NF.	`malicious_nf`	proved	auth_notif	checked
A03	The subscriptionId a consumer accepts in the 201 Created response is one the NWDAF really created and assigned in that run; a response built by the network attacker is rejected.	`network`	proved	sub_a	checked
A03	When the NWDAF accepts a cancellation for a consumer, that consumer really requested the deletion of that subscriptionId.	`malicious_nf`	attack (depends_on_ambiguity)	del_a	
A04	A subscription creation run exists, with a subscriptionId the NWDAF assigns for that run; distinct runs are distinct sessions.		reachable	ctl_bcreate	
A04	A subscription creation with its own subscriptionId is reached.	`network`	reachable	ctl_bcreate	
A04	A subscription creation with its own subscriptionId is reached, including one made by a malicious NF in its own name.	`malicious_nf`	reachable	ctl_bcreate	
R02	A notification is produced, delivered and accepted, so the retransmission question of this target is asked of a reached run.		reachable	reach_notif	
R02	A notification is produced, delivered and accepted.	`network`	reachable	reach_notif	
R02	A notification is produced, delivered and accepted for a consumer the attacker does not control.	`malicious_nf`	reachable	reach_notif	
S05	A notification is sent for a subscription that is later cancelled, so the post-cancellation question is asked of a reached run.		reachable	ctl_payload	

target	claim	attacker	answer	query	control
S05	A notification is sent for a subscription that is then cancelled.	`network`	reachable	ctl_payload	
S05	A notification is sent for a subscription that is then cancelled, for a consumer the attacker does not control.	`malicious_nf`	reachable	ctl_payload	
S04	A subscription is created and a notification for it is accepted, so the update-then-notify question is asked of a reached run.		reachable	reach_notif	
S04	A subscription is created and a notification for it is accepted.	`network`	reachable	reach_notif	
S04	A subscription is created and a notification for it is accepted, for a consumer the attacker does not control.	`malicious_nf`	reachable	reach_notif	
K02	The analytics payload of a notification sent to a consumer is not learnable by the network attacker; the control shows that a payload is known to the attacker when it belongs to the NF the attacker controls.	`network`	proved	payload_secret	checked
K02	Another legitimate NF, holding its own credentials and able to invoke the API, cannot read the analytics payload of a notification addressed to a different consumer.	`malicious_nf`	proved	payload_secret	none
K02	Another legitimate NF holding its own credentials cannot read the analytics payload of a notification addressed to a different consumer.	`malicious_nf`	proved	payload_secret	checked
Z03.10	An Unsubscribe the NWDAF accepts was preceded by a token issued to the same NF that presents it: a token issued to one NF cannot be used by another to act on the API.	`malicious_nf`	proved	auth_del	checked
Z03.10	The subscriptionId and notificationURI a consumer operates on belong to that consumer; no token moves a right over another consumer's subscription.	`malicious_nf`	attack (depends_on_ambiguity)	own_del	
Z03.10	An Unsubscribe accepted by the NWDAF is preceded by a token issued to the same NF.	`network`	proved	auth_del	checked
U01	A notification is only produced for a subscriptionId the NWDAF created for that same consumer, so another NF's notification cannot reach a consumer under that consumer's subscriptionId.	`malicious_nf`	proved	auth_notif	checked
U01	Every notification is produced for a subscription created for that same consumer; the network attacker cannot forge one.	`network`	proved	auth_notif	checked
U01	A notification is only produced for a subscriptionId the NWDAF created for that same consumer.	`malicious_nf`	proved	auth_notif	checked
U01	The consumer accepts a notification only for a subscription whose 201 it authenticated itself; the recorded counterexample comes from the model's single store of both parties and does not describe the protocol.	`malicious_nf`	attack (model_artifact)	notif_own	

target	claim	attacker	answer	query	control
U02	A notification is accepted for a live subscription, so the stale-notification question is asked of a reached run.		reachable	reach_notif	
U02	A notification is accepted for a live subscription of the consumer.	`network`	reachable	reach_notif	
U02	A notification is accepted for a live subscription of a consumer the attacker does not control.	`malicious_nf`	reachable	reach_notif	

Appendix D: How the analysis was run

An LLM analyst builds the model of the protocol from the documents and puts each property to ProVerif; every result in section 3 is an answer of the prover.

ProVerif+ Agent session session-20261114T234020, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 69, prover 4 s, wall 16 min, cost 0.129 USD.

ProVerif: 78 proofs run, 78 decided, 0 without an answer, 2 s, 38 answers recorded.

What was analysed

Protocol	NWDAF analytics subscription lifecycle
Version	19.8.0
Analysed	One NF and the NWDAF, with fixed non-UE aggregate network performance analytics; creation, update, notification and cancellation of a single subscription, with at most two notifications, excluding transfer, roaming and multiple NWDAFs.
Not analysed	The symbolic model does not prove the computational or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive control. Fixed to Rel-19. The external network attacker and the malicious legitimate NF holding its own credentials are distinguished separately; TLS authenticated encryption is modelled per endpoint, and a legitimate NF cannot read another NF's TLS session. OAuth2 subject/audience/scope/expiry and TLS identity binding follow the attached baseline; a service scope is not automatically equal to resource ownership. Monotonic tables cannot model real deletion, and the finite-state lifecycle must be explicitly delimited. The target is the specified API sub-flow, not full 5G security. This time only the granularity of the selection of the source text was corrected: the original full API chapter included all analytics types and exceeded the context, so the original protocol scope and the readability of the complete manuscript are retained; numeric payloads that were not selected remain opaque, and their numerical correctness is not verified.
Source text	609 numbered passages (498330 bytes) from the sections listed below. A reference such as s0001 in this report is one of these passages

The sections of the documents the analysis is built from:

Document	Section	Title
main	4.2.2.2	Nnwdaf_EventsSubscription_Subscribe service operation
main	4.2.2.2.1	General
main	4.2.2.2.2	Subscription for event notifications
main	4.2.2.2.3	Update subscription for event notifications
main	4.2.2.3	Nnwdaf_EventsSubscription_Unsubscribe service operation
main	4.2.2.3.1	General
main	4.2.2.3.2	Unsubscribe from event notifications
main	4.2.2.4	Nnwdaf_EventsSubscription_Notify service operation
main	4.2.2.4.1	General
main	4.2.2.4.2	Notification about subscribed event
main	5.1.1	Introduction

Document	Section	Title
main	5.1.2	Usage of HTTP
main	5.1.2.1	General
main	5.1.2.2	HTTP standard headers
main	5.1.2.2.1	General
main	5.1.2.2.2	Content type
main	5.1.2.3	HTTP custom headers
main	5.1.3.1	Resource Structure
main	5.1.3.2	Resource: NWDAF Events Subscriptions
main	5.1.3.2.1	Description
main	5.1.3.2.2	Resource definition
main	5.1.3.2.3	Resource Standard Methods
main	5.1.3.2.3.1	POST
main	5.1.3.2.4	Resource Custom Operations
main	5.1.3.3	Resource: Individual NWDAF Event Subscription
main	5.1.3.3.1	Description
main	5.1.3.3.2	Resource definition
main	5.1.3.3.3	Resource Standard Methods
main	5.1.3.3.3.1	DELETE
main	5.1.3.3.3.2	PUT
main	5.1.3.3.4	Resource Custom Operations
main	5.1.5	Notifications
main	5.1.5.1	General
main	5.1.5.2	Event Notification
main	5.1.5.2.1	Description
main	5.1.5.2.2	Operation Definition
main	5.1.6.2.2	Type NnwdafEventsSubscription
main	5.1.6.2.3	Type EventSubscription
main	5.1.6.2.4	Type NnwdafEventsSubscriptionNotification
main	5.1.6.2.5	Type EventNotification
main	5.1.6.2.7	Type EventReportingRequirement
main	5.1.6.2.22	Type NetworkPerfRequirement
main	5.1.6.2.23	Type NetworkPerfInfo
main	5.1.6.2.30	Type ThresholdLevel
main	5.1.6.2.49	Type ConsumerNfInformation
main	5.1.6.3.3	Enumeration: NotificationMethod
main	5.1.6.3.4	Enumeration: NwdafEvent

Document	Section	Title
main	5.1.6.3.10	Enumeration: NetworkPerfType
main	5.1.6.3.26	Enumeration: TermCause
main	5.1.7	Error handling
main	5.1.7.1	General
main	5.1.7.2	Protocol Errors
main	5.1.7.3	Application Errors
main	5.1.8	Feature negotiation
main	5.1.9	Security

The purpose of the protocol according to where the source step located it in the documents: an NF and NWDAF, fixed non-UE aggregated network performance analytics; create/update/notify/cancel a single subscription, at most two notifications, excluding transfer/roaming/multiple NWDAF.

The properties against the documents

Before a query was written, the analyst set the drafted list of properties against what the documents themselves say the protocol is for. The name, what a property is about and what the documents say are the analyst's reading; the places named exist in the documents.

ID	Name	About	What the documents say	Where
Z01	OAuth2 authorization of every operation	The NWDAF is assured that every subscribe, update, unsubscribe and notification it accepts from an NF service consumer was authorized by the NRF (authorization server) for the Nnwdaf_EventsSubscription API on behalf of that consumer; and, where local policy requires it, that the target UE has consented to the analytics collection.	5.1.9 says access may be authorized by OAuth2 Client Credentials with the NRF as authorization server and defines the scope nnwdaf-eventssubscription; 4.2.2.2.2 (s0287) requires the NWDAF to check user consent for targeted UEs when local policy requires it and to refuse with USER_CONSENT_NOT_GRANTED.	ts-33-501:2883, ts-33-501:2885, ts-33-501:2889, s0565, s0566, s0568, s0570, s0287
Z03	Subscription bound to its consumer	The NWDAF is assured, and the NF service consumer that created a subscription is assured, that only that consumer can update or delete the subscription resource and that notifications go only to the notificationURI the subscription currently binds.	The documents define the subscription resource, its {subscriptionId} and the notificationURI (s0303 NOTE 1 lets the notificationURI be modified to send subsequent notifications to a new consumer, s0324/s0325 delete by subscriptionId) and require OAuth2 authorization with a scope, but state no rule that the caller must be the consumer that created the subscription.	ts-33-501:2875, rfc6749:2879, s0303, s0324, s0325, s0568, s0570
D01	Notification accepted only from its NWDAF	The NF service consumer is assured that every event notification it accepts was sent by the NWDAF it subscribed to, carries an event identifier it subscribed to, and was not modified on the way.	4.2.2.4.2 has the NWDAF POST the notification to the notificationURI received at subscribe (s0340), the consumer store it and answer 204 (s0388..s0390); 5.1.9/s0565 and TS 33.501 require the SBI request/response to be mutually authenticated and SBI signalling to be integrity protected.	ts-33-501:2871, ts-33-501:2875, s0340, s0383, s0388, s0390, s0565

ID	Name	About	What the documents say	Where
A03	Consumer and NWDAF agree on subscription	The NF service consumer and the NWDAF hold the same subscription after creation, update and cancellation: the subscriptionId the consumer accepted is the one the NWDAF created, and a cancellation the NWDAF accepted was requested by that consumer.	4.2.2.2.2 s0278..s0282 (create, assign subscriptionId, respond 201 with the representation and Location), 4.2.2.2.3 s0300..s0306 (update), 4.2.2.3.2 s0324..s0327 (delete, respond 204).	s0278, s0280, s0282, s0300, s0306, s0324, s0327
A04	Distinct subscriptionId per creation	Each successful subscription creation gets its own subscriptionId and resource, and a replayed create or delete request cannot add a subscription instance or delete another consumer's subscription.	s0280 'assign an event subscriptionId', s0281 store the subscription, s0325..s0327 remove the corresponding subscription on DELETE; the documents state no idempotency or replay rule of their own.	ts-33-501:2875, s0278, s0280, s0281, s0325, s0327
R02	Each request and notification accepted once	The NWDAF and the NF service consumer accept a given create, update or delete request, and a given notification, at most once, so that a retransmission produces no duplicate subscription, duplicate update or duplicate stored notification.	The service operations are plain HTTP POST/PUT/DELETE with 201/200/204 responses (s0282, s0304..s0306, s0325..s0327) and the notification is a POST answered 204 (s0388..s0390); the documents give no transaction identifier or deduplication rule.	s0282, s0306, s0327, s0388, s0390
S05	No notification after cancellation	After the NWDAF has removed a subscription on DELETE, or has terminated it with a termCause, that subscriptionId produces no further notification and accepts no further update.	4.2.2.3.2 s0325..s0327 (remove the subscription), s0386 (termCause: the NWDAF indicates it will send no further notifications for it), s0283 (muting stores events until retrieval).	s0325, s0327, s0386
S04	Updated subscription used for later notifications	After an update, later notifications follow the updated subscription parameters: the notificationURI the subscription now binds, and the muting state, without falling back to a replaced earlier context.	4.2.2.2.3 s0300..s0303: the NWDAF updates and stores the subscription, and NOTE 1 says the notificationURI can be modified so that subsequent notifications go to a new consumer.	s0298, s0300, s0303, s0306
K02	Analytics payload only to its subscriber	The analytics carried in an event notification is kept from every party other than the NF service consumer that subscribed to it and the NWDAF that produced it, including other legitimate NFs.	5.1.9 puts the API under OAuth2-authorized SBI access and TS 33.501 requires SBI signalling confidentiality; the notification body carries the analytics in eventNotifications (s0343..s0367).	ts-33-501:2871, s0343, s0340, s0565
Z03.10	Token subject and subscription ownership	The NWDAF is assured that the OAuth2 token presented for an operation was issued to the NF that presents it, and that the subscriptionId and notificationURI it operates on belong to that NF, so that no token or subscription transfer moves a right or a notification to another consumer.	5.1.9 names the grant, the authorization server and the scopes (s0565..s0570) but the source text states no rule binding a subscription to the consumer that created it; RFC 6749 10.3 notes that OAuth2 itself gives the resource server no method to check that a presented access token was issued to the presenting client.	rfc6749:2879, rfc6749:2827, s0565, s0568, s0570, s0303
U01	Notifications do not cross subscriptions	The NF service consumer accepts a notification only for a subscription of its own: the subscriptionId and event identifier in the notification belong to a subscription it created, and the notification is not one produced for another consumer.	s0383: the notification carries the event subscriptionId; s0340: it is sent to the notificationURI the subscription binds; s0388..s0390: the consumer stores it and answers 204.	s0340, s0383, s0385, s0388

ID	Name	About	What the documents say	Where
U02	No stale notifications after update or cancel	After an update or cancellation has taken effect, the NF service consumer accepts no notification generated from the old subscription parameters; only a notification already in flight when the change took effect may be accepted.	s0300..s0306 (update stored, response sent) and s0325..s0327 (subscription removed, 204) state when a change takes effect; the documents state no rule for notifications in flight across that instant.	s0300, s0306, s0325, s0327, s0386

The analyst added no property: the drafted list covers what the documents state.

The security statements of the documents, paragraph by paragraph

The host listed the 398 paragraphs of the sections in which the documents say what the protocol protects and against what, and the analyst said of each what it made of it: a property, an attacker, or why neither. Taken up: 24. Not taken: 374. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

Paragraphs	Where	Taken up as	Why not
P001 to P002	rfc6749:2778-2786	No goal stated	The introduction points to the OAuth threat model; it states no requirement of its own.
P003 to P006	rfc6749:2790-2823	Z01, Z03.10, `malicious_nf`	
P007 to P010	rfc6749:2827-2850	Z03, Z03.10, `malicious_nf`	
P011 to P014	rfc6749:2859-2877	Z01, K02, `network`, `malicious_nf`	
P015	rfc6749:2879-2881	Z03.10, Z03, `malicious_nf`	
P016 to P021	rfc6749:2885-2917	Outside the scope	Refresh tokens; the API uses the Client Credentials grant only (s0565), which has none.
P022 to P029	rfc6749:2921-2980	Outside the scope	Authorization codes and redirection URIs of the authorization-code grant; not used by this API.
P030 to P033	rfc6749:2984-3003	Outside the scope	Resource owner password credentials grant; not used by this API.
P034 to P035	rfc6749:3013-3019	Z01, K02, `network`	
P036	rfc6749:3023-3028	D01, `network`	
P037 to P039	rfc6749:3032-3041	The prover cannot be asked	Guessing client credentials is a matter of entropy and rate limiting, which the prover cannot count.
P040 to P062	rfc6749:3045-3210	Outside the scope	Phishing, cross-site request forgery, clickjacking, input validation, open redirectors and the implicit grant: the consumers here are network functions, not browsers or resource owners, and the API uses the Client Credentials grant without user agents.
P063	ts-33-501:2491-2491	Outside the scope	Bidding down of 5G access security features between UE and network, not the SBI analytics API.
P064 to P074	ts-33-501:2495-2517	Outside the scope	UE/SUPI authentication, serving network authorization and 5G key lengths; the selected flow has no UE and no access security.

Paragraphs	Where	Taken up as	Why not
P075 to P125	ts-33-501:2523-2631	Outside the scope	Requirements on the UE, AS/NAS confidentiality and integrity, subscription credential storage and subscriber privacy; the selected flow is analysed as non-UE aggregate analytics and the task leaves UE data out.
P126 to P178	ts-33-501:2637-2761	Outside the scope	Requirements on 5GC/NG-RAN keys, the gNB, the ng-eNB and F1/E1; not the SBI analytics API.
P179 to P198	ts-33-501:2767-2811	Outside the scope	Signalling protection on the access network (RRC/NAS) and the SEAF; the SBI signalling of this API is not what these paragraphs speak of.
P199 to P213	ts-33-501:2819-2853	Outside the scope	Requirements on UDM, AUSF and UDR subscription data; user consent data is outside the selected flow, which carries no UE identifier.
P214	ts-33-501:2859-2859	No goal stated	Assumption about operator trust zones on which the requirements that follow rest.
P215 to P217	ts-33-501:2865-2869	No goal stated	Confidentiality, integrity, replay protection and topology hiding for NF discovery and registration; this API is service consumption, not discovery.
P218 to P220	ts-33-501:2871-2875	Z01, D01, A04, `network`, `malicious_nf`	
P221 to P222	ts-33-501:2879-2881	No goal stated	Description of the NRF role in discovery and in issuing tokens.
P223 to P227	ts-33-501:2883-2891	Z01, Z03.10, `malicious_nf`	
P228 to P241	ts-33-501:2895-2923	Outside the scope	NEF requirements and SCP requirements; the selected flow is direct communication between one consumer and one NWDAF without SCP or NEF.
P242 to P254	ts-33-501:2929-2953	Outside the scope	End-to-end core network interconnection security for roaming intermediaries (IPX, SEPP); the task excludes roaming exchange.
P255 to P314	ts-33-501:2957-3081	Outside the scope	SEPP, N32-c/N32-f, protection of attributes across PLMNs and IPUPS; the selected flow is inside one PLMN.
P315 to P318	ts-33-501:3085-3091	No goal stated	Originating PLMN/SNPN header; advice for routing between networks, not a property of this API.
P319 to P337	ts-33-501:3097-3141	No goal stated	Signalling monitoring, security visibility and configurability of the operator deployment.
P338 to P369	ts-33-501:3149-3215	Outside the scope	Algorithm identifiers and algorithm selection for AS/NAS protection; the model treats the primitives as ideal and the task excludes computational and quantum resistance.
P370 to P375	ts-33-501:3219-3231	Outside the scope	5G-RG and NSSAAF requirements; not this API.
P376 to P380	ts-33-501:6540-6548	Outside the scope	Security on the N3 user plane interface.
P381 to P394	ts-33-501:9656-10917	Outside the scope	EAP-TLS attachment and IAB security requirements.
P395 to P398	ts-33-501:11439-11901	Outside the scope	MBS and federated-learning NEF requirements; not this API.

The parties and the protected values, one by one

The host listed the 22 parties and protected values that the drafted properties name, and the analyst said of each which attacker controls the party or holds the value, or why no such attacker was set. With an attacker: 15. Without: 7. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	NF service consumer	party	`malicious_nf`	
S02	NWDAF	party	Everything lost by definition	The NWDAF is the party whose authorization and notification behaviour every target here is about. If it is dishonest, the properties it assures are void by definition, and the documents require none of them under a dishonest NWDAF.
S03	NRF	party	Everything lost by definition	The NRF is the authorization server; with it every authorization property is lost by definition.
S04	UDM	party	Outside the scope	User consent data concerns UE-targeted analytics; the selected flow is non-UE aggregate network performance and carries no SUPI or GPSI.
S05	Other legitimate NFs	party	`malicious_nf`	
S06	Other NF consumers	party	`malicious_nf`	
S07	OAuth2 access token	value	`malicious_nf`	
S08	scope nnwdaf-eventssubscription	value	`malicious_nf`	
S09	scope nnwdaf-eventssubscription:transfer	value	Outside the scope	The nnwdaf-eventssubscription:transfer scope covers analytics subscription transfer, which the task excludes.
S10	User consent subscription data	value	Outside the scope	User consent subscription data is retrieved over Nudm_SDM; the selected flow has no UE and the task leaves the UDM interaction out.
S11	subscriptionId	value	`malicious_nf`	
S12	notificationURI	value	`malicious_nf`	
S13	notifCorrId	value	Nothing gained	The notification correlation identifier is an opaque value carried inside the notification; it authorizes nothing and is only meaningful to one consumer.
S14	Event notification	value	`malicious_nf`	
S15	eventSubscriptions	value	`malicious_nf`	
S16	Subscription resource	value	`malicious_nf`	
S17	Subscription request	value	`malicious_nf`	
S18	Subscription state	value	`malicious_nf`	
S19	Notification	value	`malicious_nf`	
S20	notifFlag state	value	Outside the scope	The notifFlag muting and retrieval state belongs to the EneNA feature, which the selected flow does not model.
S21	Event notification payload	value	`malicious_nf`	
S22	Analytics data	value	`malicious_nf`	

The properties under each attacker

The attackers are the analyst's, set before a query was written; each answer was recorded under one of them. A cell is the prover's answer for the property under that attacker. "No answer" means an attacker set for the property under which the prover left a claim undecided or nothing was recorded. An empty cell is an attacker the analyst did not set for the property.

Attacker	Has	Basis	Where
`network`	the network only: it reads, drops, changes and sends messages, and holds no secret of a party		
`malicious_nf`	Its own NF client credentials and a valid OAuth2 access token with the nnwdaf-eventssubscription scope from the NRF, so it can invoke Subscribe, Update, Unsubscribe and answer notifications as itself. It can keep, replay and reorder every message it receives, and it may learn a subscriptionId or notificationURI of another consumer from wherever its own role exposes it.	The task scope separates an external network attacker from a malicious legitimate NF holding its own credentials. 5.1.9 grants the single scope nnwdaf-eventssubscription to every authorized consumer (s0565..s0570), and TS 33.501 5.9.2.1 requires only that the NRF authorize the NF, not that the NWDAF tie a resource to its creator.	ts-33-501:2883, ts-33-501:2875, s0565, s0568, s0570

ID	`network`	`malicious_nf`
Z01	Holds	Holds
Z03	No answer	Fails
D01	Holds	Holds
A03	Holds	Fails
A04	No answer	No answer
R02	No answer	No answer
S05	No answer	No answer
S04	No answer	No answer
K02	Holds	Holds
Z03.10	Holds	Fails
U01	Holds	Holds
U02	No answer	No answer

Source and targets

Source: 609 units, 498330 bytes, frozen in source-delivery-01.json. Targets: 11 approved (Z01, Z03, D01, A03, A04, R02, S05, S04, K02, U01, U02); the questions asked: 3. The list of targets was approved as drafted, beforehand, with --approve-targets.

Decisions the host took by its documented default (task `decisions: auto`), for you to check:

- approve_package: approved source package protocol-9d033e9fb7e29d6bc9a30e01de6b

This step as it reports itself: 2 requests (2 failed), 0.000 USD, 422144 tokens in and 11079 out, 286 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
50	Z01, Z03, Z03.10, D01, A03, A04, R02, S05, S04, K02, U01, U02	plain_primitives	The SBI (mutual TLS) is modelled per endpoint with ideal signatures and ideal public-key encryption, and the OAuth2 token as a signed triple. The prover treats these as perfect, so no computational or quantum resistance is claimed. Every answer about authorization, notification integrity and payload secrecy rests on this idealisation. Otherwise the tokens and keys are faithful to the flow.

The analyst's closing note

Twelve targets were answered on model.pv, a model of the subscribe, notify and unsubscribe flow with the SBI protected per endpoint (signatures, encryption to the recipient, every payload bound to its recipient), an OAuth2 token issued by the NRF and checked by the NWDAF, an honest consumer and a second NF whose credentials the attacker holds.

Established by proof, each with a control that shows the property can fail: no operation is accepted without a token issued to the same NF (Z01, also for the malicious-NF attacker; the consent half of Z01 does not arise because the flow is non-UE aggregate); notifications are authenticated, bound to the subscribing consumer and confidential against both attackers (D01, K02, U01); the consumer's 201 is authentic (A03); and a token issued to one NF cannot be used by another (Z03.10).

One weakness found, answers under Z03, Z03.10 and A03: the NWDAF does not tie a subscription to the consumer that created it, so another NF with a valid token for the same scope can unsubscribe a subscriptionId it did not create, and the NWDAF will also remove a subscription the named consumer never asked to delete. The source states no ownership rule (s0278, s0324..s0327), and RFC 6749 notes that OAuth2 gives the resource server no way to check who a token was issued to. The finding is marked unclear in novelty because TS 33.501 only requires each NF to validate incoming messages. The recommendation — record the token subject with the resource and compare it on Update and Unsubscribe — was checked on a copy of the model: the recorded attack is gone, the normal run is still found, and the other thirteen recorded answers are unchanged.

What blocks the rest: five targets (A04, R02, S05, S04, U02) are answered by reachability only, because ProVerif's tables only grow and therefore cannot represent the removal or replacement of a subscription, the instant an update or a cancellation takes effect, or a message count. The next step is a bounded model that unrolls the consumer and the NWDAF to three steps with the subscription state carried in the chain, followed by a state review of the replay, repeat-one-time-action, desynchronisation and no-rollback patterns. Two model caveats are recorded in notes.md: the model publishes the second NF's key, so every "network" answer is really an answer against an attacker with one NF's credentials, and PUT is not modelled, so the redirection of the notificationURI with the same missing binding is argued from s0303 rather than shown.