Security analysis of ADRF data retrieval subscription

This report in another language: 中文.

19.7.0 · ProVerif+ Agent report

Documents	5G System; Analytics Data Repository Services; Stage 3 (protocol definition)
Properties examined	13: 6 hold, 5 fail (5 attacks), 2 without an answer
Findings	2

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 Nadrf_DataManagement API lets an NF service consumer ask the ADRF, the analytics data repository function, for analytics or data the ADRF already stores, and for the notifications that follow when new records arrive. Three parties take part: the NF service consumer that subscribes, the ADRF that stores the records and sends the notifications, and the NRF that acts as OAuth 2.0 authorization server and issues the access tokens.

The consumer sends a NadrfDataRetrievalSubscription with a notifCorrId, a notificationURI, a time window and one analytics or data subscription. The ADRF creates the subscription, assigns a subscriptionId, answers 201 Created with a Location header, and later posts a NadrfDataRetrievalNotification to the notificationURI with the notifCorrId, a time stamp and the analytics, the data or a fetch instruction. The consumer stores each notification and answers 204 No Content; a DELETE on the subscription removes it.

The parties:

- **NF service consumer**: subscribes and receives notifications. Holds: NF identity, long-term credentials, callback address (notificationURI).
- **ADRF**: creates the subscription, sends notifications, removes it. Holds: stored analytics records, subscription store.
- **NRF**: issues the OAuth 2.0 access token. Holds: authorization server role.

Figure 1 shows the exchange as the documents specify it.

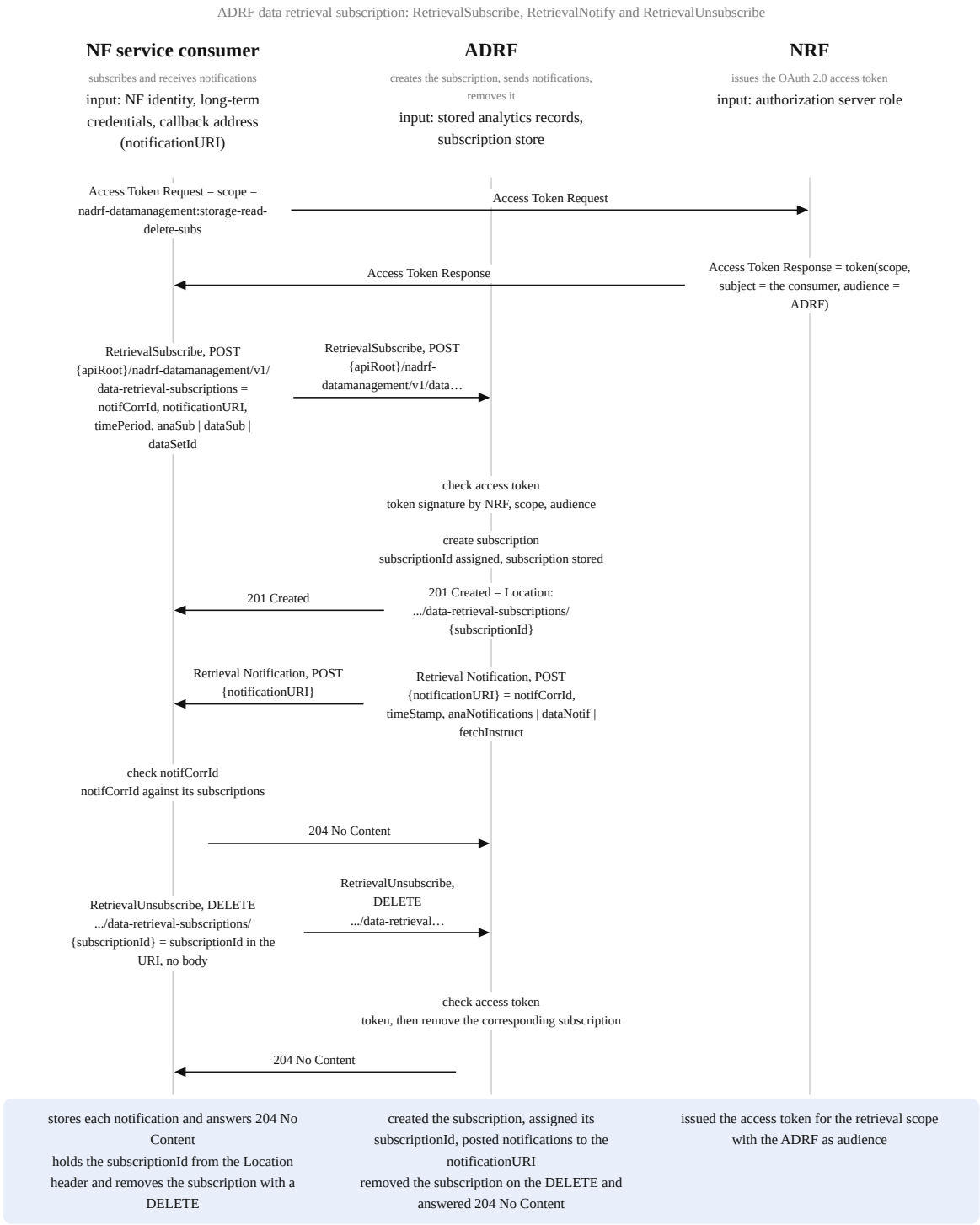


Figure 1. The protocol as specified.

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 2 findings: what follows is not stated in the documents. For 2 of them it is unclear whether the documents state them.

The subscription exchange itself holds up: the consumer accepts only the values it sent, the ADRF accepts only NF identities the NRF authorized, and the notification payload reaches no other NF. Two weaknesses were found. A notification identifies its subscription only by a notifCorrId that must be unique per consumer and by the callback address the subscriber names, so one NF can have its notifications accepted by another NF's endpoint as that NF's own. And a retrieval subscription can be deleted by any NF holding the retrieval scope once it can name the subscriptionId, because the documents give no ownership rule for the DELETE. All thirteen targets have an answer; two of them rest on a model that idealises the transport and the subscription lifetime, so the deletion condition and the notification ordering stay caveated.

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 Any NF with the retrieval scope can delete another NF's retrieval subscription

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	Z03, Z05 (section 3)
Source text	s0021, s0032, s0377, s0200
Shown by	ProVerif attack query `z03_owner`; Figures 2 and 3

What happens. Any NF that holds the API scope can remove any retrieval subscription of any other NF, because the DELETE carries only the subscriptionId and no body and the ADRF has no rule to compare the requester with the NF that created the subscription. The attacker sends the DELETE with its own valid token and with the subscriptionId of the victim's subscription; the ADRF removes that subscription and answers 204 No Content to the attacker. The victim is never told that its subscription has gone.

What the attacker needs. The adversary is a legitimate NF service consumer with its own credentials and an access token carrying the scope "nadrf-datamanagement:storage-read-delete-subs", which the documents allow for every consumer of the API. It also needs the subscriptionId of the victim's subscription, which the documents require neither to be secret nor to be unguessable.

What the attacker gains. The victim's retrieval subscription stops without its knowledge: it receives no further analytics it subscribed to, and the adversary can repeat this on any subscription whose identifier it can name, denying the analytics service to chosen NFs.

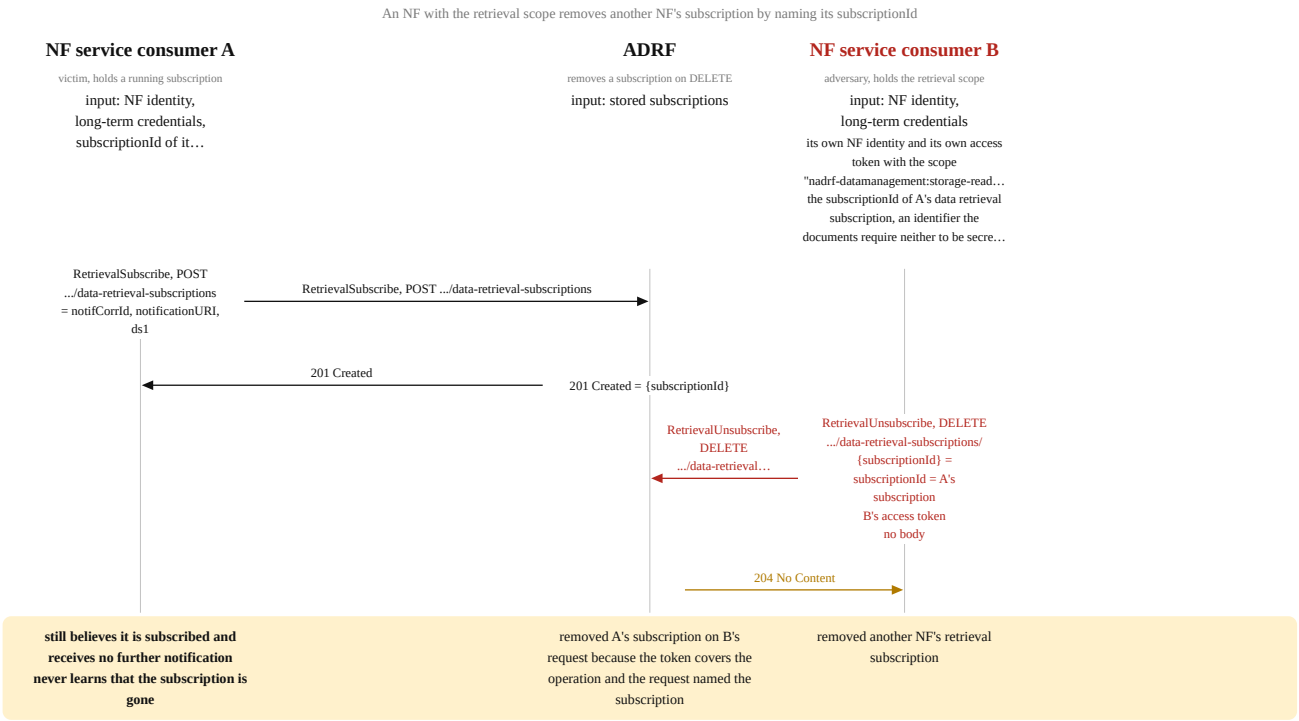


Figure 2. An NF with the retrieval scope removes another NF's subscription by naming its subscriptionId. Property Z03; attack query z03_owner.

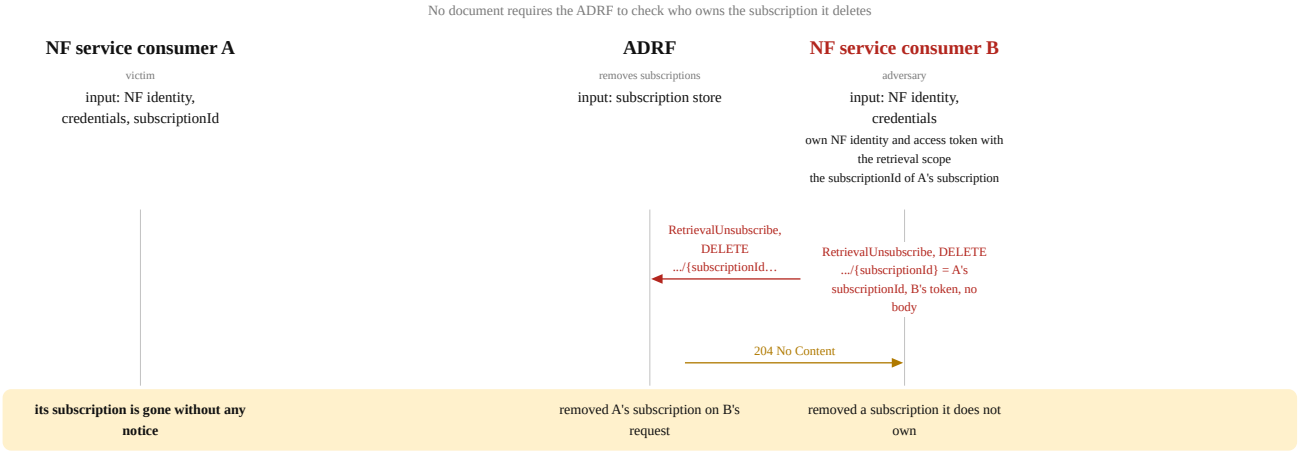


Figure 3. No document requires the ADRF to check who owns the subscription it deletes. Property Z05; attack query z03_owner.

Proposed change. Store the requesting NF's identity with the subscription when it is created, and on the DELETE of an Individual ADRF Data Retrieval Subscription reject the request with 403 Forbidden unless the requester is the NF that created that subscriptionId. Alternatively define a resource-level scope per subscription so that holding the retrieval scope does not authorize the removal of another consumer's subscription.

Check of the change with ProVerif. Not checked: The change is a check the ADRF makes and could be modelled, but the session ended before a copy of the model with the ownership check could be run, so the recommendation is given unchecked.

2.2 A notification made for one NF's subscription is accepted by another NF's endpoint

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (ts-29-500:6152)
Value	medium
Properties	D01, A08, U01 (section 3)
Source text	s0325, s0328, s0045, s0057
Shown by	ProVerif attack query `d01_own`; Figures 4, 5 and 6

What happens. A notification carries the notifCorrId chosen by the subscriber but no subscriptionId and no identity of the subscriber, and the ADRF posts it to the notificationURI the subscription named. An NF that names another NF's callback address as its own notificationURI and uses the same notifCorrId has its notifications delivered to that other NF, whose endpoint matches them by notifCorrId and accepts them as the notifications of its own subscription. The receiving NF then holds analytics of a data set it did not subscribe to.

What the attacker needs. The adversary is a legitimate NF service consumer with its own credentials and token, and it needs two things the documents do not protect: the callback address of another NF, which is a reachable network address, and the notifCorrId value that NF uses, which only has to be unique per consumer and need not be unguessable.

What the attacker gains. The consumer stores and acts on analytics about a data set it did not subscribe to, and any automated decision it takes on those notifications is taken on another NF's data. It cannot tell those notifications from its own, and it answers 204 No Content to the ADRF, which records the notification as delivered.

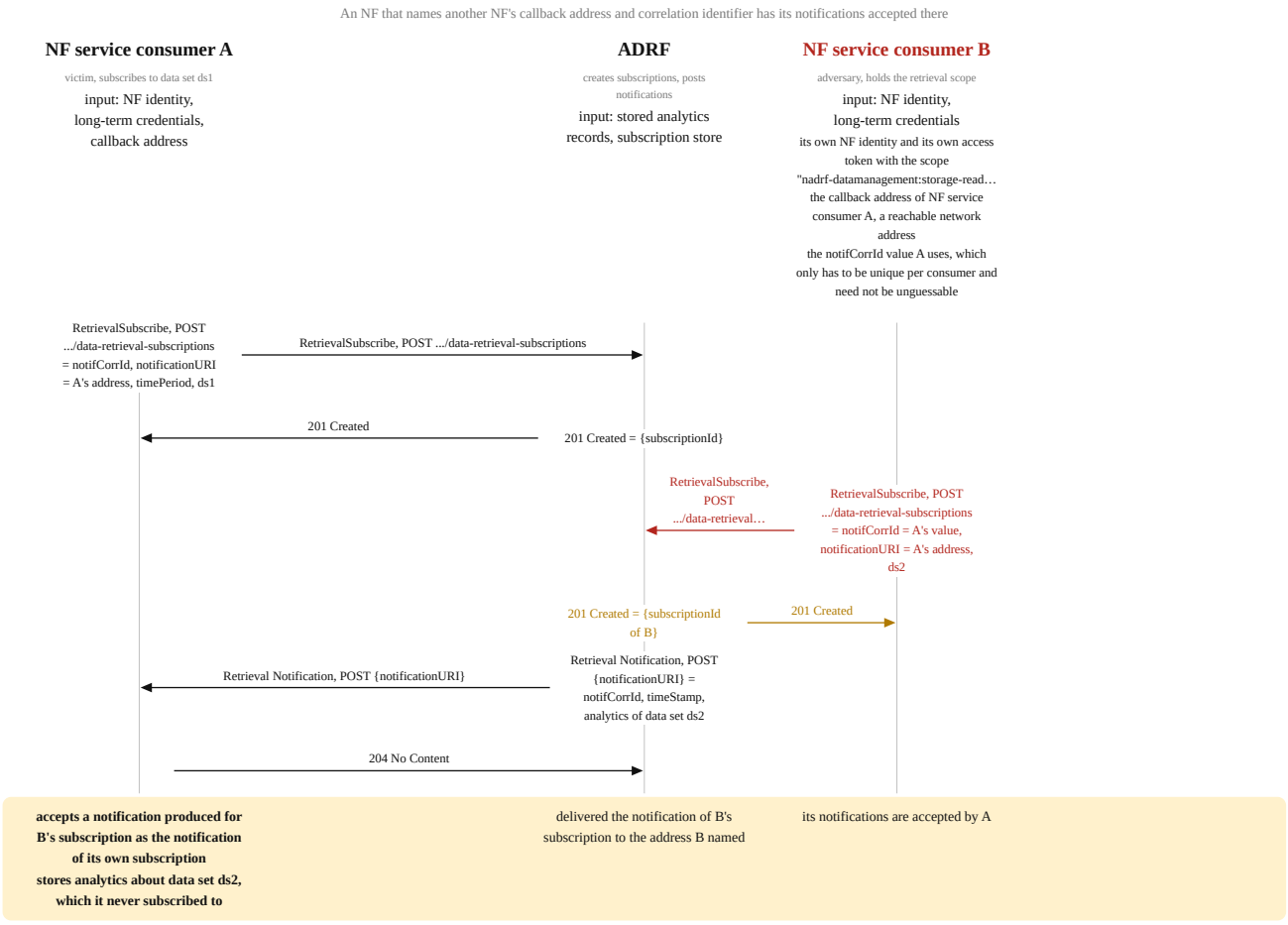


Figure 4. An NF that names another NF's callback address and correlation identifier has its notifications accepted there. Property D01; attack query d01_own.

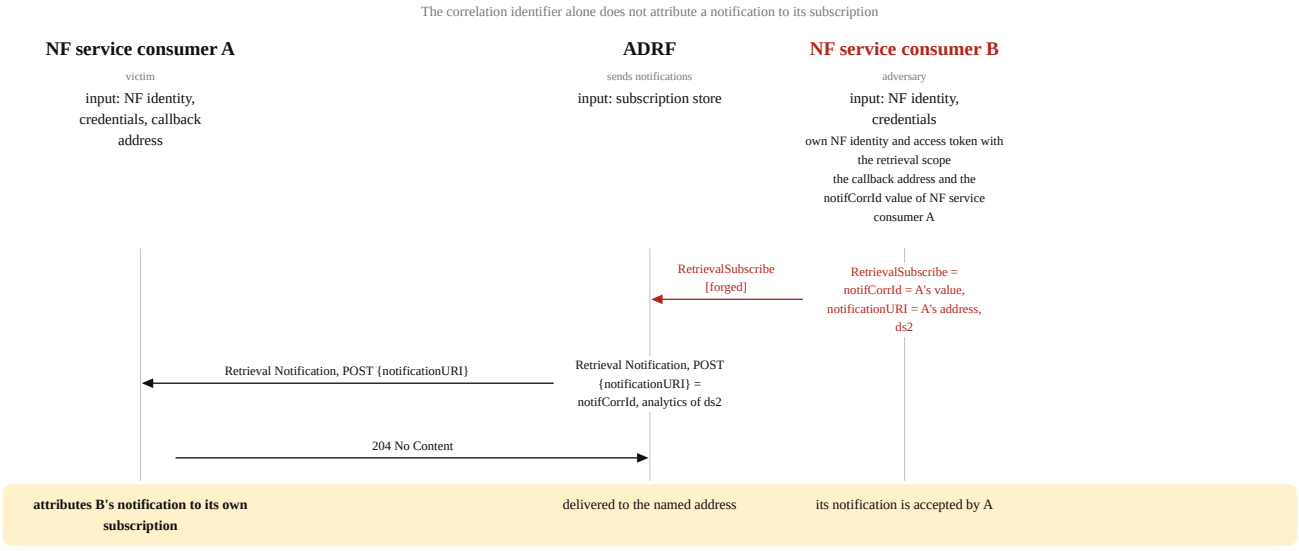


Figure 5. The correlation identifier alone does not attribute a notification to its subscription. Property A08; attack query d01_own.

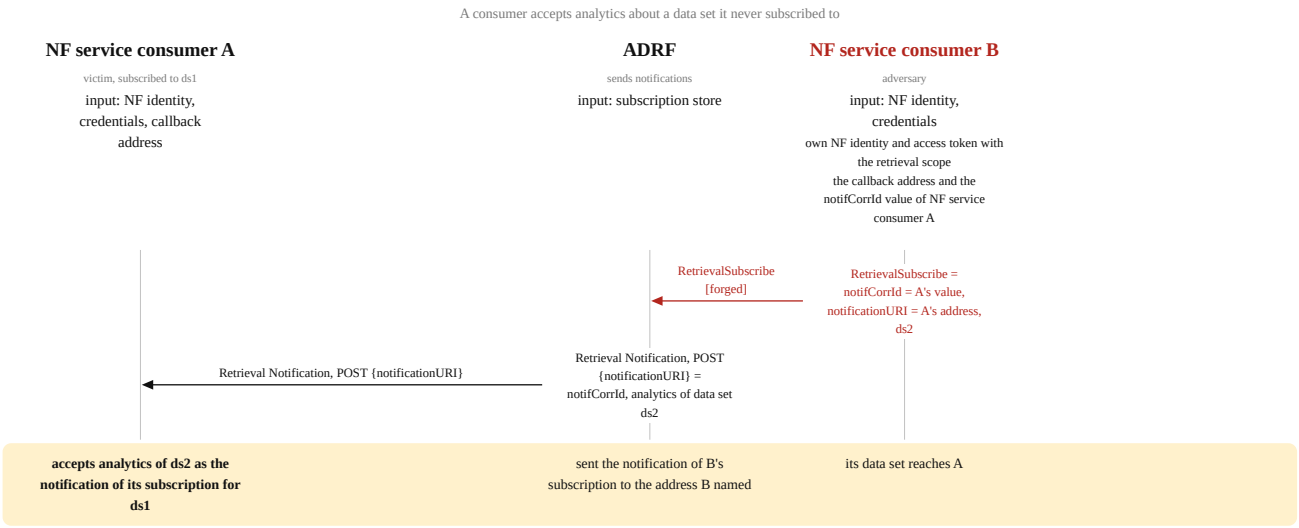


Figure 6. A consumer accepts analytics about a data set it never subscribed to. Property U01; attack query d01_own.

Proposed change. Carry the subscriptionId in the NadrfDataRetrievalNotification and require the consumer to accept a notification only when that identifier is one of its own subscriptions, or bind the notificationURI to the subscribing NF at subscription time and reject a notificationURI that belongs to another NF.

Check of the change with ProVerif. Not checked: The change is a field of the notification plus a check at the consumer, so it can be modelled; it was not, because the session ended before a copy of the model with the subscriptionId in the notification payload could be run, and the recommendation is therefore given unchecked.

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
A03	**Subscription request matches created subscription.** Every time the consumer accepts a subscription creation confirmation (201 with Location), there is a real run of an ADRF whose received subscription request has a notifCorrId, notificationURI and subscription content matching the consumer's local values; every time an ADRF accepts a subscription request, there is a real run of a consumer with that ADRF as its peer.	no compromise	Holds	medium	yes
A04	**Subscription and notification are injective.** Every time an ADRF accepts a subscription request, it corresponds to a distinct real subscription run of the consumer (the same notifCorrId is not accepted twice as two subscriptions); every time the consumer accepts a notification, it corresponds to a distinct notification sent by the ADRF for that subscription.	no compromise	Holds	low	unclear
D01	**Notification comes from the ADRF.** Every time the consumer accepts a Retrieval Notification, there is a real notification sent by that ADRF for that subscription of that consumer, the notifCorrId matches the subscription, and the notification payload has not been replaced.	no compromise	Fails (1 attack)	medium	unclear

ID	Property	Compromise assumed	Result	Value	In the documents
A08	**Notification belongs to its own subscription.** For every notification the consumer accepts, its notifCorrId and notificationURI match the values the consumer previously submitted for that subscription, and the notification is not attributed to another subscription.	no compromise	Fails (1 attack)	medium	unclear
Z01	**Operations require a valid access token.** Every time an ADRF submits a subscription creation or cancellation operation, and every time the consumer accepts a notification, there must exist a valid grant issued by the authorization server that covers that operation (scope and audience match).	no compromise	Holds	medium	yes
Z03	**Only the subscription owner can remove it.** Every time a cancellation operation is submitted for a subscriptionId, the subject, audience and resource of its grant correspond to the consumer that created that subscription; the grant cannot be used for another subscription or another operation.	no compromise	Fails (1 attack)	medium	unclear
S05	**No notifications after unsubscribe.** After the ADRF successfully processes a cancellation (or sends a terminationReq), no new notification is produced for that subscriptionId, and the old subscriptionId cannot trigger the deletion side effect again.	no compromise	Reachable only	low	unclear
R02	**One notification accepted at most once.** The same logical notification instance (same notifCorrId and same content) is accepted at most once at the consumer.	no compromise	No answer	none	unclear
K02	**Notification payload stays with its consumer.** The notification payload and the data/analysis retrieved with fetchInstruct cannot be derived by any entity other than the subscribing consumer and the ADRF (including other legitimate NFs).	no compromise	Holds	medium	yes
K01	**Access token cannot be derived.** The attacker cannot derive a valid OAuth2 token issued by the authorization server (including scope and audience).	no compromise	Holds	low	yes
U01	**Notification and data set stay with the subscriber.** The notificationURI and dataStoreId of a notification match the values submitted by the consumer that created the subscription, and the notification is not attributed to another subscription.	no compromise	Fails (1 attack)	medium	unclear
U02	**Notification is authentic and the subscription is ended.** The notification comes from the subscription's ADRF and belongs to that subscription; after cancellation the subscription no longer produces notifications.	no compromise	Holds	medium	yes
Z05	**Cross-NF access policy has a normative basis.** The question asks for the normative basis of the cross-NF data access policy, not for a decidable trace property.	no compromise	Fails (1 attack)	medium	unclear

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Binding of the subscriber to dataStoreId/notificationURI	No: the notificationURI is chosen by the subscriber when it subscribes and the ADRF never compares it with the subscriber's own identity, and the notification carries neither a subscriptionId nor the subscriber's identity, so a consumer cannot check that a notification belongs to the subscription it created. The only binding is the notifCorrId, which the source requires to be unique only per consumer and never to be secret.	A08, U01, D01

Question	Answer	Rests on
Q2: Notification authenticity, cross-subscription replacement and finite-state cancellation	Notifications are authentic in origin, unmodified and carry the notifCorrId the consumer chose, and their payload reaches no other NF; but the attribution of a notification to a subscription is not protected, and the removal of a subscription is not bound to its creator, so the finite-state cancellation rule rests on the ADRF's own check of its removal marker, which this model does not settle.	D01, U02, S05, Z03
Q3: Data access policy for other legitimate NFs must have a normative basis	There is no such normative basis: the only rule the documents give is the OAuth 2.0 scope set, in which a single resource/operation-level scope covers the retrieval subscription, unsubscribe and notify operations for every consumer that holds it. TS 29.501 4.10 requires that such a scope-only design must not grant undue delete access, which is a requirement on the API designer that the analysed API does not carry into a per-subscription rule.	Z05, Z03

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:

- The transport is modelled as a signature by the sender plus encryption to the receiver, not as TLS sessions: message protection is therefore transferable in the model, which is compensated by binding the recipient address inside the signed notification. The real protocol blocks such a relay by mutual TLS.
- The subscriptionId space of the model holds a single value, so the claim that one request is not accepted as two subscriptions is weak; whether a random identifier would prevent the deletion of another NF's subscription is not settled here.
- The subscription lifetime is a monotonic table plus a removal marker: the ordering claim that no notification is sent after the ADRF removed the subscription is not proved, only that the ADRF checks the marker before notifying.
- The consumer is modelled as one run, so whether one notification is stored twice is not decided; ProVerif returned no answer for that query.
- Only the branch without SUPI or GPSI is analysed; the user consent check with the UDM required for data about a UE (s0004) is outside the model.
- The external data collection and the fetch of data with Nadrf_DataManagement_RetrievalRequest (s0052, s0062) are outside the analysed sub-procedure, so the protection of fetched data is not answered.
- The OAuth 2.0 token is modelled as a signature by the NRF over scope, subject and audience; expiry, revocation and the token request procedure of TS 29.510 are not modelled.
- Timing, availability, resource exhaustion and the computational strength of the primitives are not askable and are not covered.

What the host records:

- No answer for: S05, R02.
- The prover gave no answer (Appendix C): R02: r02_once. The property was asked and the prover neither proved it nor found an attack, so the property is shown without an answer.
- Set by the analyst and without a proof or an attack (Appendix D): S05 under `network`.
- Proved without a control query (Appendix C): A03: a03_created, A03: a03_sub, A04: a04_sub, D01: d01_origin, A08: d01_origin, Z01: a03_sub, Z01: z01_unsub, Z03: z01_unsub, K02: k02_secret, K01: a03_sub, U01:

d01_origin, U02: d01_origin, Z05: a03_sub. 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.

A03: Subscription request matches created subscription

Every time the consumer accepts a subscription creation confirmation (201 with Location), there is a real run of an ADRF whose received subscription request has a notifCorrId, notificationURI and subscription content matching the consumer's local values; every time an ADRF accepts a subscription request, there is a real run of a consumer with that ADRF as its peer.

The consumer's 201 Created can only carry the subscription values it sent, so it cannot be made to accept a subscription created for another request, and the ADRF accepts a subscription only from an NF to which the NRF had issued the retrieval token. Both hold against an external attacker and against an NF that holds its own valid credentials.

- Result: Holds; answers what is asked: yes; usable
- Value medium: The two directions of the subscription exchange are the basis of everything the consumer later receives, and the result is a proof over a transport that authenticates the NF identity; the caveat is that the transport is idealised.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2871
- Rests on: Transport is modelled as a sender signature plus encryption to the receiver, not as TLS sessions.; The consumer runs one subscription; a second subscription of the same consumer is not modelled.
- Next step: Nothing for this claim; the notification side of the same target is answered by D01.

A04: Subscription and notification are injective

One subscription request is accepted as one subscription, shown for a single subscription identifier. The second half of the target, that one notification is accepted once, is not settled: the consumer is modelled as a single run and the query that would show it gave no answer.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: The answer rests on a degenerate identifier space and a single-run consumer, so it says little about the protocol.
- Stated in the documents: unclear
- Rests on: The subscriptionId space of the model is a single value, so the injectivity claim is weak.; The consumer is one process, so repeated acceptance cannot be observed.
- Open: Whether a repeated notification is stored twice by a consumer.; Whether a repeated subscription request from one NF is accepted twice.
- Next step: A bounded model with a real identifier space and a consumer that keeps the notifications it has already stored.

D01: Notification comes from the ADRF

Every time a consumer accepts a Retrieval Notification, a real notification was sent by that ADRF for that consumer's subscription, the notifCorrId matches the subscription, and the notification payload has not been replaced.

An accepted notification is always an authentic ADRF notification unmodified and with the correlation identifier the consumer chose, but it need not have been produced for one of that consumer's subscriptions: an NF that names the consumer's callback address and its notifCorrId makes the ADRF deliver its own notifications there, and the consumer accepts them as its own.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: It isolates a specified gap: the notification has no identifier of the subscription it was produced for, and the source relies on a value required to be unique only per consumer.
- Stated in the documents: unclear. Source text it rests on: ts-29-500:6152
- Rests on: The attacker knows the victim's callback address (public) and its notifCorrId value, which the source requires only to be unique per consumer.; Recipient binding inside the ADRF signature stands in for the mutual TLS binding of the real transport.
- Open: Whether a notification can be bound to its subscription at all without a subscriptionId in the body.
- Next step: Add the subscriptionId to the notification in a model and re-run these queries to see whether the attack is removed.

A08: Notification belongs to its own subscription

Every notification a consumer accepts has a notifCorrId and notificationURI that correspond to the values the consumer previously submitted for that subscription, and the notification is not attributed to another subscription.

The correlation identifier is not enough to attribute a notification to a subscription: because it has to be unique only for a given consumer and the notification has no other identifier, an accepted notification can belong to another consumer's subscription, and analytics about a data set the consumer never asked for are accepted as its own.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: It shows what the correlation identifier the source designs can and cannot do, with the exact conditions and the step where the misattribution happens.
- Stated in the documents: unclear
- Rests on: The attacker knows the victim's notifCorrId value; a randomly chosen correlation identifier would prevent it, which the source does not require.; The consumer matches by notifCorrId alone, as s0328 suggests for a common callback URI.
- Next step: Check whether carrying the subscriptionId in the notification removes the attack.

Z01: Operations require a valid access token

Every subscription creation or cancellation the ADRF submits, and every notification a consumer accepts, must be covered by a valid grant (matching scope and audience) issued by the authorization server.

Every subscription and every removal the ADRF accepts is covered by a token the NRF issued to that NF for the retrieval scope with the ADRF as audience, so no consumer can act with a token that was not issued to it. What the token cannot do is distinguish which subscription a consumer may act on, which is the authorization question the other targets pursue.

- Result: Holds; answers what is asked: yes; usable with caveats

- Value medium: It shows that the OAuth 2.0 check does what it can do and where it stops, which is what the RFC 6749 warning about token-to-client binding says.
- Stated in the documents: yes. Where the documents say it: rfc6749:2879
- Rests on: The token is modelled as an NRF signature over scope, subject and audience; expiry and revocation are not modelled.; The scope set of the API is modelled as one scope covering the retrieval operations.
- Next step: Define a resource-level scope per subscription and ask the same queries.

Z03: Only the subscription owner can remove it

Every cancellation submitted with a subscriptionId has a grant whose subject, audience and resource correspond to the consumer that created the subscription; the grant cannot be used for another subscription or another operation.

An NF that holds the retrieval scope can remove any retrieval subscription whose subscriptionId it can name: the DELETE carries nothing but that identifier and the ADRF has no ownership rule. The consequence is the silent loss of the subscription for its owner, and it needs no more than a legitimate NF with its own token.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value medium: It turns the requirement of TS 29.501 4.10 into a concrete failure for this API, with the exact condition on the identifier.
- Stated in the documents: unclear. Source text it rests on: ts-29-501:1801
- Rests on: The attacker can name the victim's subscriptionId; the source requires it to be unguessable nowhere.; The model gives the ADRF no ownership check, because the source specifies none.
- Next step: Re-run with an ownership check in the ADRF to confirm that the attack is removed.

S05: No notifications after unsubscribe

After the ADRF successfully processes a cancellation (or issues a terminationReq), no new notification is produced for that subscriptionId, and the old subscriptionId cannot trigger the deletion side effect again.

The ADRF checks its removal marker before every notification, so in this model a notification is never produced for a subscription it has already removed; the recorded falsification comes from an order-free query that lets a notification appear before the removal, which the protocol allows. The ordering property itself is not settled here.

- Result: Reachable only; answers what is asked: partly; usable with caveats
- Value low: The target's property is not decided; only the presence of the ADRF check is shown.
- Stated in the documents: unclear
- Rests on: Subscription removal is a marker in a monotonic table, not a real deletion.; The ordering of removal and notification is not expressible in this model.
- Open: No notification after the ADRF removed the subscription, as an ordering claim.
- Next step: A bounded model that unrolls the subscription lifetime and shows that a notification for a removed subscription is never accepted.

R02: One notification accepted at most once

The same logical notification instance (same notifCorrId and same content) is accepted at most once by the consumer.

ProVerif could neither prove nor refute that a consumer accepts at most one notification payload per subscription; the derivation it produced is an over-approximation, not a trace. The protocol itself has no replay detection in the notification, so the property depends on the consumer and on the transport.

- Result: No answer; answers what is asked: undetermined; not usable
- Value none: No answer was obtained, so nothing about the protocol is established for this target.
- Stated in the documents: unclear
- Rests on: The consumer is modelled as one run, so repetition cannot be observed.; Notification replay over the transport is prevented by the modelled signature and encryption.
- Open: Whether one notification is stored twice, which ProVerif could not decide here.
- Next step: A model with a replicated consumer and a stored-notifications table.

K02: Notification payload stays with its consumer

The notification payload and the data/analytics retrieved by fetchInstruct cannot be derived by any entity other than the subscription's consumer and the ADRF (including other legitimate NFs).

No entity other than the ADRF and the NF the notification is delivered to obtains the notified analytics, including a second legitimate NF that holds its own credentials. The protection is the transport, so it holds only as long as the notification is bound to the address it is delivered to, which is what the model's signature over the recipient represents.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: A confidentiality proof for the notification body that holds against an attacker with its own NF credentials, but it depends on the transport abstraction.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2871
- Rests on: Payload confidentiality rests entirely on the modelled transport; the documents give no end-to-end protection for the notification body.; The fetch instructions and the retrieval of data with Nadrf_DataManagement_RetrievalRequest are outside the model.
- Next step: Nothing for the notification body; the fetch path would need the retrieval request operation in the model.

K01: Access token cannot be derived

The attacker cannot derive a valid OAuth2 token issued by the authorization server (including scope and audience).

No operation is accepted with a token the NRF did not issue to that NF, so neither the network attacker nor another NF can forge or derive a valid token; the only tokens in play are the ones the NRF issued to their subjects.

- Result: Holds; answers what is asked: yes; usable
- Value low: It confirms what the OAuth 2.0 requirement already states, with no new weakness.
- Stated in the documents: yes. Where the documents say it: rfc6749:2870
- Rests on: The token format is idealised as an unforgeable NRF signature.; Only unforgeability is shown; the unguessability of a token value is a property of the primitive, not of this model.
- Next step: Nothing.

U01: Notification and data set stay with the subscriber

The notification's notificationURI and dataStoreId match the values submitted by the consumer that created the subscription, and are not attributed to any other subscription.

The consumer can be made to accept, as the notification of its own subscription, analytics about a data set it never subscribed to, because the notification identifies the subscription only by a correlation identifier that another NF may share, and carries no identifier of the data set it asked for. It holds that the notification is delivered to the address the consumer named, which is what makes the misattribution possible.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: It ties the misattribution to the concrete attribute the consumer asked for, which is what an editor would need to see.
- Stated in the documents: unclear
- Rests on: The consumer matches notifications by notifCorrId only, and the correlation value need not be unguessable.; The data set identifier is not carried in the notification, which is what the source specifies.
- Next step: Carry the subscriptionId in the notification and re-run.

U02: Notification is authentic and the subscription is ended

The notification comes from the subscription's ADRF and belongs to that subscription; after cancellation, that subscription produces no further notifications.

Every accepted notification is an authentic, unmodified ADRF notification with the correlation identifier the consumer chose, so origin and integrity hold even against a malicious NF. What the consumer cannot tell is whether the notification was produced for its own subscription, and the fact that acceptance stops after the removal is only because its own run has ended.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The origin and integrity part is a solid proof over the modelled transport; the second half is not decided.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2871
- Rests on: The consumer is one run, so the end of the subscription is not tested as a state change.; Recipient binding inside the signature stands in for mutual TLS.
- Open: The ordering half of the target, that no notification arrives after the removal.
- Next step: A bounded model of the subscription lifetime.

Z05: Cross-NF access policy has a normative basis

The question asks for the normative basis of cross-NF data access policy, not for some decidable trace property.

The only rule the documents give is the OAuth 2.0 scope, and it does not distinguish the consumers of a subscription: any NF with the retrieval scope may invoke unsubscribe on any subscription it can name. So the analysed documents carry no normative basis for a cross-NF restriction beyond the scope itself, and the model shows what that leaves open.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: It answers the governance question the user asked and points at the paragraph that requires the API designer to check it.
- Stated in the documents: unclear. Source text it rests on: ts-29-501:1801

- Rests on: The answer is a reading of the documents plus the deletion attack, not a trace property of its own.; Reading the stored analytics through Nadrf_DataManagement_RetrievalRequest is outside the model.
- Open: Which NF may read the stored analytics through the retrieval request operation, which the task leaves out.
- Next step: Extend the model to the retrieval request operation to answer the read side as well.

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
A03	proved		a03_created			3 proved
A04	proved		a04_sub			2 proved, 1 reachable
D01	proved		d01_origin		d01_own (holds_in_source)	2 proved, 1 attack
A08	proved		d01_origin		d01_own (holds_in_source)	1 proved, 1 attack
Z01	proved		a03_sub			3 proved
Z03	proved		z01_unsub		z03_owner (depends_on_ambiguity)	1 proved, 1 attack
S05	reachable		s05_reach		s05_none (model_artifact)	1 reachable, 1 attack
R02	reachable		k02_reach			1 reachable, 2 undecided
K02	proved		k02_secret			2 proved
K01	proved		a03_sub			2 proved
U01	proved		d01_origin		d01_own (holds_in_source)	1 proved, 1 attack
U02	proved		d01_origin			2 proved, 1 reachable
Z05	proved		a03_sub		z03_owner (holds_in_source)	1 proved, 1 attack

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
A03	Every 201 Created the consumer accepts carries the notifCorrId, notificationURI, data set and subscriptionId of a subscription request that consumer itself sent.	`network`	proved	a03_created	none
A03	Every 201 Created the consumer accepts carries the notifCorrId, notificationURI, data set and subscriptionId of a subscription request that consumer itself sent, also when another NF holds its own credentials.	`malicious_nf`	proved	a03_created	none
A03	Every subscription the ADRF accepts was requested by an NF to which the NRF had issued an access token with the retrieval scope.	`malicious_nf`	proved	a03_sub	none
A04	The same subscription request (same subscriber, notifCorrId, notificationURI and data set) is not accepted as two different subscriptions.	`malicious_nf`	proved	a04_sub	none
A04	A subscription request is accepted by the ADRF, which is the situation the second half of the target speaks about.	`network`	reachable	k02_reach	
A04	The same subscription request is accepted as one subscription.	`network`	proved	a04_sub	none
D01	Every notification the consumer accepts was sent by the ADRF, for the same notifCorrId and the same payload, and the payload is unmodified.	`malicious_nf`	proved	d01_origin	none

target	claim	attacker	answer	query	control
D01	Every notification the consumer accepts was sent by the ADRF for one of that consumer's own subscriptions.	`malicious_nf`	attack (holds_in_source)	d01_own	
D01	Every accepted notification was sent by the ADRF with the notifCorrId the consumer chose and an unmodified payload.	`network`	proved	d01_origin	none
A08	An accepted notification is attributed only to the subscription the consumer created, never to another consumer's subscription.	`malicious_nf`	attack (holds_in_source)	d01_own	
A08	The network attacker alone cannot have a notification accepted by a consumer: every accepted notification is an authentic ADRF notification with the correlation identifier the consumer chose.	`network`	proved	d01_origin	none
Z01	Every retrieval subscription the ADRF accepts is covered by an access token that the NRF issued to the requesting NF for the retrieval scope and for this ADRF as audience.	`malicious_nf`	proved	a03_sub	none
Z01	Every removal of a retrieval subscription the ADRF accepts is covered by an access token that the NRF issued to the requesting NF for the retrieval scope and for this ADRF as audience.	`malicious_nf`	proved	z01_unsub	none
Z01	Every subscription the ADRF accepts is covered by a token the NRF issued to the requesting NF.	`network`	proved	a03_sub	none
Z03	The NF that removes a retrieval subscription is the NF service consumer that created it.	`malicious_nf`	attack (depends_on_ambiguity)	z03_owner	
Z03	The ADRF removes a retrieval subscription and answers 204 No Content only after a request the network attacker cannot make, since the request must carry a token it cannot obtain.	`network`	proved	z01_unsub	none
S05	The ADRF sends a Retrieval Notification for a stored subscription to its notificationURI.	`network`	reachable	s05_reach	
S05	A removed subscription produces no further notification.	`network`	attack (model_artifact)	s05_none	
R02	The consumer accepts a Retrieval Notification for its subscription, so the situation the target speaks about is reached.	`network`	reachable	k02_reach	
R02	A consumer accepts at most one notification payload per subscription, so a notification instance is not acted upon twice.	`network`	undecided	r02_once	
R02	A consumer accepts at most one notification payload per subscription, also when another NF holds its own credentials.	`malicious_nf`	undecided	r02_once	
K02	The payload of a notification reaches no entity other than the ADRF and the NF that the notification is delivered to; no other NF and not the network attacker learns it.	`malicious_nf`	proved	k02_secret	none
K02	No entity other than the ADRF and the notified NF learns the payload of a notification.	`network`	proved	k02_secret	none

target	claim	attacker	answer	query	control
K01	No operation is accepted with a token that the authorization server did not issue to that NF, so no attacker and no other NF can derive or forge a valid token.	`malicious_nf`	proved	a03_sub	none
K01	No operation is accepted with a token the authorization server did not issue to that NF.	`network`	proved	a03_sub	none
U01	An accepted notification reports the data set the consumer subscribed to, and its notificationURI is the one of its own subscription.	`malicious_nf`	attack (holds_in_source)	d01_own	
U01	The notification the consumer accepts is delivered to the notificationURI it named, and its payload reaches no other entity.	`network`	proved	d01_origin	none
U02	Every notification the consumer accepts is an authentic notification of the ADRF, unmodified and carrying the notifCorrId the consumer chose.	`malicious_nf`	proved	d01_origin	none
U02	The consumer removes its subscription after accepting a notification, so the removal the second half of the target speaks about is reached.	`network`	reachable	run	
U02	Every accepted notification is an authentic ADRF notification and the consumer removes its subscription afterwards.	`network`	proved	d01_origin	none
Z05	There is a normative rule in the analysed documents that keeps one NF service consumer from removing another's retrieval subscription; the analysis shows what the documents' only authorization rule allows.	`malicious_nf`	attack (holds_in_source)	z03_owner	
Z05	The network attacker alone can neither subscribe nor remove a subscription, because both operations need a token the NRF issued to an NF.	`network`	proved	a03_sub	none

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-20261114T231523, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 53, prover 4 s, wall 18 min, cost 0.114 USD.

ProVerif: 69 proofs run, 67 decided, 2 without an answer, 2 s, 30 answers recorded.

What was analysed

Protocol	ADRF data retrieval subscription
Version	19.7.0
Analysed	Fixed stored non-UE aggregated analytics records; RetrievalSubscribe/Notify/Unsubscribe, omitting external data collection. Explicitly, only the branch without SUPI/GPSI is chosen; this cannot be generalised to mean no user consent is required.
Not analysed	The symbolic model does not prove computational security of the algorithms or resistance to quantum attacks; attack traces hold only under explicit assumptions and are not directly the same as specification vulnerabilities. Each 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 does not automatically equal resource ownership. A monotone table cannot model real deletion, and the finite-state lifecycle must be clearly delimited. The target is the specified API sub-flow, not complete 5G security. The native Word table of the previous delivery had cross-clause localisation errors; this time the same scope was re-parsed from the unchanged manuscript and independently verified. The old model is an artifact pending review; the old mismatched sources cannot be taken as the specification or used directly to claim the original conclusions are false.
Source text	447 numbered passages (331824 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	4.2.2.6	Nadrf_DataManagement_RetrievalSubscribe service operation
main	4.2.2.6.1	General
main	4.2.2.6.2	Requesting retrieval and subscription of data or analytics
main	4.2.2.7	Nadrf_DataManagement_RetrievalUnsubscribe service operation
main	4.2.2.7.1	General
main	4.2.2.7.2	Requesting removal of retrieval subscription for data or analytics
main	4.2.2.8	Nadrf_DataManagement_RetrievalNotify service operation
main	4.2.2.8.1	General
main	4.2.2.8.2	Notification about subscribed data or analytics
main	4.2.2.8.3	Notification about data or analytics that are about to be deleted
main	5.1	Nadrf_DataManagement Service API

Document	Section	Title
main	5.1.1	Introduction
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	Resources
main	5.1.3.1	Overview
main	5.1.3.2	Resource: ADRF Data Store Records
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	5.1.3.2.3.2	GET
main	5.1.3.2.4	Resource Custom Operations
main	5.1.3.3	Resource: Individual ADRF Data Store Record
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	5.1.3.3.4	Resource Custom Operations
main	5.1.3.4	Resource: ADRF Data Retrieval Subscriptions
main	5.1.3.4.1	Description
main	5.1.3.4.2	Resource Definition
main	5.1.3.4.3	Resource Standard Methods
main	5.1.3.4.3.1	POST
main	5.1.3.4.4	Resource Custom Operations
main	5.1.3.5	Resource: Individual ADRF Data Retrieval Subscription
main	5.1.3.5.1	Description
main	5.1.3.5.2	Resource Definition
main	5.1.3.5.3	Resource Standard Methods
main	5.1.3.5.3.1	DELETE
main	5.1.3.5.4	Resource Custom Operations
main	5.1.4	Custom Operations without associated resources
main	5.1.4.1	Overview

Document	Section	Title
main	5.1.4.2	Operation: request-storage-sub
main	5.1.4.2.1	Description
main	5.1.4.2.2	Operation Definition
main	5.1.4.3	Operation: request-storage-sub-removal
main	5.1.4.3.1	Description
main	5.1.4.3.2	Operation Definition
main	5.1.4.4	Operation: remove-stored-data-analytics
main	5.1.4.4.1	Description
main	5.1.4.4.2	Operation Definition
main	5.1.5	Notifications
main	5.1.5.1	General
main	5.1.5.2	Retrieval Notification
main	5.1.5.2.1	Description
main	5.1.5.2.2	Target URI
main	5.1.5.2.3	Standard Methods
main	5.1.5.2.3.1	POST
main	5.1.5.3	ADRF Alert Notification
main	5.1.5.3.1	Description
main	5.1.5.3.2	Target URI
main	5.1.5.3.3	Standard Methods
main	5.1.5.3.3.1	POST
main	5.1.6	Data Model
main	5.1.6.1	General
main	5.1.6.2	Structured data types
main	5.1.6.2.1	Introduction
main	5.1.6.2.2	Type: NadrfDataStoreRecord
main	5.1.6.2.3	Type: NadrfDataStoreSubscription
main	5.1.6.2.4	Type: NadrfDataRetrievalSubscription
main	5.1.6.2.5	Type: NadrfDataRetrievalNotification
main	5.1.6.2.6	Type: NadrfDataStoreSubscriptionRef
main	5.1.6.2.7	Type: NadrfStoredDataSpec
main	5.1.6.2.8	Type: DataSubscription
main	5.1.6.2.10	Type: StorageHandlingInfo
main	5.1.6.2.11	Type: NadrfAlertNotification
main	5.1.6.2.12	Type: NadrfAlertNotificationResponse
main	5.1.6.2.13	Type: DataSetTag

Document	Section	Title
main	5.1.6.3	Simple data types and enumerations
main	5.1.6.4	Data types describing alternative data types or combinations of data types
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
main	5.1.6.2.9	

This is how the source step located the purpose of the protocol in the documents: to store non-UE aggregated analytics records in a fixed way; RetrievalSubscribe/Notify/Unsubscribe, omitting external data collection. It explicitly selects only the branch without SUPI/GPSI and must not be generalized to mean that user consent is not required.

The properties against the documents

Before writing a query, the analyst checked 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 all the analyst's reading; the places named do exist in the documents.

ID	Name	About	What the documents say	Where
A03	Subscription request matches created subscription	The NF service consumer and the ADRF: the consumer is assured that the notifCorrId, notificationURI and subscription content it sent are those of the 201 Created it accepts, and the ADRF that the request it accepts comes from an NF that the authorization server authorized for this operation.	The consumer sends the NadrfDataRetrievalSubscription with notifCorrId, notificationURI, timePeriod and one of anaSub/dataSub/dataSetId (s0009-s0016); the ADRF creates the subscription, assigns subscriptionId and answers 201 Created with a representation and a Location header (s0019-s0023). TS 33.501 5.9.2.1 requires mutual authentication between consumer and producer (ts-33-501:2871).	s0003, s0009, s0015, s0019, s0023
A04	Subscription and notification are injective	The ADRF against a repeated subscription request, and the consumer against a repeated notification: one request creates one subscription, one notification is accepted once.	The ADRF shall create a new subscription and assign a subscriptionId upon reception of a POST (s0019-s0022); the consumer shall store the notification and answer 204 (s0057-s0059). The documents state no anti-replay rule; uniqueness of notifCorrId is required only per subscription for a given consumer (s0325).	s0019, s0020, s0021, s0057, s0059, s0325
D01	Notification comes from the ADRF	The NF service consumer is assured that each accepted Retrieval Notification was sent by the ADRF for one of its own subscriptions, with the notifCorrId of that subscription and an unmodified payload.	The ADRF shall invoke Nadrf_DataManagement_RetrievalNotify and POST the NadrfDataRetrievalNotification, which carries notifCorrId, timeStamp and one of anaNotifications/dataNotif/fetchInstruct (s0040, s0045-s0051); the consumer stores it and answers 204 (s0057-s0059). Integrity is delegated to SBI transport security (ts-33-501:2871, s0266).	s0040, s0045, s0046, s0057, s0059

ID	Name	About	What the documents say	Where
A08	Notification belongs to its own subscription	The NF service consumer: an accepted notification is attributed only to the subscription it created, not to another subscription and not to another consumer's subscription.	notifCorrId is provided by the consumer and shall be unique per subscription for a given NF service consumer (s0325); the notification carries notifCorrId and is said to be useful when a common callback URI is used for multiple subscriptions (s0328); the notificationURI is the notification target address chosen by the consumer (s0325). RFC 6749 10.15 warns about endpoints that send credentials to an address the requester supplied (rfc6749:3154).	s0015, s0045, s0046, s0325, s0328
Z01	Operations require a valid access token	The ADRF is assured that each retrieval subscription request it accepts was authorized by the authorization server for the Nadrf_DataManagement API with the required scope; the consumer holds such a token before invoking the operation.	Access may be authorized by OAuth2 with the Client Credentials grant, the NRF acting as authorization server; the consumer shall obtain a token before consuming the service (s0372-s0374); the scopes are "nadrf-datamanagement" and "nadrf-datamanagement:storage-read-delete-subs" (s0375-s0377). RFC 6749 10.3 states that the resource server has no means to check that a token was issued to the client presenting it (rfc6749:2879).	rfc6749:2879, s0372, s0373, s0375, s0377
Z03	Only the subscription owner can remove it	The NF service consumer that created a retrieval subscription: another NF service consumer, even one holding a valid access token with the retrieval scope, cannot remove that subscription.	The DELETE on the Individual ADRF Data Retrieval Subscription carries only the subscriptionId in the URI and no body (s0032, s0205-s0208); the only authorization the documents give for the operation is the API scope, which covers the RetrievalUnsubscribe operation as a whole (s0377); TS 29.501 4.10 states that a service-level-only scope grants all operations to all consumers allowed to access the service and warns that this must not grant undue delete access (ts-29-501:1801).	ts-29-501:1801, s0032, s0033, s0377, s0200
S05	No notifications after unsubscribe	The NF service consumer: after the ADRF has removed the subscription (204 No Content) or has requested termination, no further notification for that subscription arrives.	Upon a DELETE the ADRF shall remove the corresponding subscription and answer 204 (s0033-s0035); terminationReq indicates that no further notifications related to this subscription will be provided (s0328).	s0027, s0033, s0034, s0035, s0328
R02	One notification accepted at most once	The NF service consumer against a repeated delivery of the same Retrieval Notification: the same notification instance is not stored or acted upon twice.	The consumer shall store the notification and answer 204 (s0057-s0059); the notification carries timeStamp but the documents state no replay detection rule (s0046, s0047).	s0045, s0046, s0047, s0057, s0059
K02	Notification payload stays with its consumer	The notified data or analytics (anaNotifications, dataNotif, fetchInstruct) is kept from every entity other than the ADRF and the subscribed NF service consumer, including other legitimate NFs and the network attacker.	The notification body carries the analytics, the data or the fetch instructions (s0045-s0051); the documents give no end-to-end protection and rely on SBI transport integrity and confidentiality (s0266, ts-33-501:2871).	s0045, s0049, s0050, s0051, s0062
K01	Access token cannot be derived	The authorization server's token: neither the network attacker nor another NF can derive a valid token it was not issued.	The consumer shall obtain a token from the NRF before consuming the service (s0372-s0374); RFC 6749 10.3 requires that tokens cannot be generated, modified or guessed by unauthorised parties (rfc6749:2870), and 10.10 that guessing is prevented (rfc6749:3032).	rfc6749:2870, s0372, s0373

ID	Name	About	What the documents say	Where
U01	Notification and data set stay with the subscriber	The NF service consumer: the notificationURI and the requested data set of a notification are those of the subscription it created, and the notification is not attributed to another subscription.	notificationURI is mandatory in the subscription and is the notification target address (s0325); the ADRF POSTs to the received notificationURI (s0045); dataSetId selects the stored data or analytics records (s0325, s0014).	s0014, s0015, s0045, s0325
U02	Notification is authentic and the subscription is ended	The NF service consumer: an accepted notification was produced by the ADRF for its subscription, and once the subscription is removed no notification of that subscription is accepted.	The ADRF sends the notification to the notificationURI (s0045); the consumer stores it (s0057); a DELETE removes the subscription (s0033-s0035); terminationReq ends further notifications (s0328).	s0045, s0057, s0033, s0034, s0328
Z05	Cross-NF access policy has a normative basis	The ADRF and the other NF service consumers: which NFs may read or delete the analytics stored in the ADRF and the subscriptions of other NFs, and where that rule comes from.	The only authorization rule the documents give for the API is the OAuth2 scope set (s0372-s0377); TS 29.501 4.10 says a service-level-only scope grants every operation to every consumer allowed to access the service and that additional scopes should be defined where access must be restricted (ts-29-501:1795, ts-29-501:1801).	ts-29-501:1795, ts-29-501:1801, s0372, s0375, s0377

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: 42. Not taken: 356. 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	Introduction to OAuth 2.0 security considerations; it defers to other documents and states no goal for this API.
P003 to P006	rfc6749:2790-2823	Z01, K01	
P007 to P010	rfc6749:2827-2850	Z01, `malicious_nf`	
P011 to P015	rfc6749:2859-2881	K01, Z01, Z03	
P016 to P021	rfc6749:2885-2917	No goal stated	Refresh tokens; this API uses the client credentials grant and the analysed flow has no refresh token.
P022 to P025	rfc6749:2921-2943	No goal stated	Authorization codes; not part of the client credentials grant used for this API.
P026 to P029	rfc6749:2947-2980	No goal stated	Redirection URI manipulation in the authorization code flow; no such flow and no redirection endpoint here.
P030 to P033	rfc6749:2984-3003	No goal stated	Resource owner password grant; the NRF issues tokens with the client credentials grant and no resource owner takes part.
P034 to P035	rfc6749:3013-3019	K01	
P036	rfc6749:3023-3028	Z01	

Paragraphs	Where	Taken up as	Why not
P037 to P039	rfc6749:3032-3041	K01	
P040 to P042	rfc6749:3045-3066	No goal stated	Phishing of end-user passwords; there is no end user and no password in this API.
P043 to P048	rfc6749:3070-3110	No goal stated	Cross-site request forgery on browser endpoints; no browser and no user-agent session here.
P049 to P050	rfc6749:3119-3138	No goal stated	Clickjacking on an authorization page; not part of this machine-to-machine API.
P051 to P052	rfc6749:3142-3150	The prover cannot be asked	Input sanitising and code injection are implementation defects; a symbolic protocol model cannot be asked about them.
P053 to P054	rfc6749:3154-3167	A08, U01, `malicious_nf`	
P055 to P062	rfc6749:3170-3210	Z01, Z03	
P063	ts-33-501:2491-2491	Outside the scope	Bidding down between UE and network over AS/NAS; the analysed API has no algorithm negotiation.
P064 to P072	ts-33-501:2495-2511	Outside the scope	Authentication and authorization of the UE and the serving network; the task fixes the non-UE, already stored analytics branch.
P073 to P074	ts-33-501:2515-2517	Outside the scope	Key lengths and algorithm dependence for AS/NAS protection; out of the analysed API.
P075 to P078	ts-33-501:2523-2529	Outside the scope	Requirements on the UE over Uu; no UE takes part in the modelled sub-procedure.
P079 to P089	ts-33-501:2533-2553	Outside the scope	User data and signalling confidentiality between UE and gNB; outside the analysed API.
P090 to P103	ts-33-501:2557-2583	Outside the scope	Integrity protection between UE and gNB; outside the analysed API.
P104 to P110	ts-33-501:2587-2599	Outside the scope	Storage of subscription credentials in the UE's tamper resistant hardware; no UE or USIM in scope.
P111 to P125	ts-33-501:2603-2631	Outside the scope	Subscriber privacy and SUPI/SUCI handling at the UE; the task fixes the branch without SUPI or GPSI.
P126	ts-33-501:2637-2637	Outside the scope	General statement about gNB requirements; no gNB in scope.
P127 to P136	ts-33-501:2641-2659	Outside the scope	Ciphering between UE and gNB; outside the analysed API.
P137 to P147	ts-33-501:2663-2683	Outside the scope	Integrity protection between UE and gNB; outside the analysed API.
P148 to P156	ts-33-501:2687-2703	Outside the scope	gNB setup and O&M; outside the analysed API.
P157 to P158	ts-33-501:2707-2709	Outside the scope	Key management inside the gNB; outside the analysed API.
P159 to P162	ts-33-501:2713-2721	Outside the scope	User plane and control plane handling at the gNB; outside the analysed API.
P163 to P168	ts-33-501:2725-2735	Outside the scope	Secure environment of the gNB; outside the analysed API.
P169 to P176	ts-33-501:2739-2755	Outside the scope	gNB F1 and E1 interfaces; outside the analysed API.

Paragraphs	Where	Taken up as	Why not
P177 to P178	ts-33-501:2759-2761	Outside the scope	Requirements on the ng-eNB; outside the analysed API.
P179 to P196	ts-33-501:2767-2805	Outside the scope	Signalling confidentiality, integrity and subscriber privacy between ng-eNB and 5GC over NG-RAN interfaces, not over the SBI API analysed here.
P197 to P198	ts-33-501:2809-2811	Outside the scope	Requirements on the SEAF; not a party of this API.
P199 to P207	ts-33-501:2819-2837	Outside the scope	Generic subscriber privacy and SUPI protection requirements on the network; the task fixes the branch without SUPI or GPSI.
P208 to P213	ts-33-501:2841-2853	Outside the scope	Requirements on AUSF and UDR for authentication data; not part of the analysed API.
P214	ts-33-501:2859-2859	No goal stated	Scope statement about operator trust zones; it states no goal and gives an attacker nothing beyond the network and its own NF credentials.
P215 to P220	ts-33-501:2865-2875	A03, D01, Z01, A04, `malicious_nf`	
P221 to P227	ts-33-501:2879-2891	Z01, K01, `malicious_nf`	
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF exposure towards Application Functions; the analysed sub-procedure has no AF and no NEF.
P235 to P241	ts-33-501:2911-2923	Outside the scope	Service Communication Proxy behaviour; the analysed procedure is direct communication between two NFs.
P242 to P254	ts-33-501:2929-2953	Outside the scope	End-to-end interconnection security for roaming; the analysed procedure stays inside one operator network and has no SEPP.
P255 to P303	ts-33-501:2957-3055	Outside the scope	SEPP and roaming intermediary requirements on N32; no SEPP and no roaming take part in the analysed procedure.
P304 to P312	ts-33-501:3059-3075	Outside the scope	Attribute protection over N32-f between SEPPs; the analysed procedure does not cross that interface.
P313 to P314	ts-33-501:3079-3081	Outside the scope	Packet filtering for GTP-U at the IPUPS; not part of the analysed API.
P315 to P318	ts-33-501:3085-3091	Outside the scope	Originating network identifier in roaming signalling; no roaming in the analysed procedure.
P319 to P327	ts-33-501:3097-3115	Outside the scope	Signalling traffic monitoring requirements; the monitoring function is not part of the analysed procedure.
P328 to P337	ts-33-501:3121-3141	Outside the scope	Visibility and configurability of AS/NAS security towards UE applications; no UE in scope.
P338 to P357	ts-33-501:3149-3189	The prover cannot be asked	Algorithm identifiers for NAS/AS ciphering and integrity; the strength of a primitive is not something a symbolic prover decides and these algorithms do not protect the analysed API.
P358 to P369	ts-33-501:3193-3215	The prover cannot be asked	Algorithm selection and bidding down between UE and network; a symbolic model cannot decide primitive strength and the analysed API has no algorithm negotiation of its own.
P370 to P375	ts-33-501:3219-3231	Outside the scope	Requirements on 5G-RG and NSSAAF; neither is a party of the analysed API.
P376 to P380	ts-33-501:6540-6548	Outside the scope	Security on the N3 interface; outside the analysed API.
P381 to P385	ts-33-501:9656-9666	Outside the scope	EAP-TLS with and without subscription identifier privacy; not part of the analysed API.

Paragraphs	Where	Taken up as	Why not
P386 to P394	ts-33-501:10893-10917	Outside the scope	IAB annex requirements; outside the analysed API.
P395 to P396	ts-33-501:11439-11443	Outside the scope	MBSF and MBSTF requirements; outside the analysed API.
P397 to P398	ts-33-501:11899-11901	Outside the scope	NEF security requirements; no NEF in the analysed procedure.

The parties and the protected values, one by one

The host listed the 21 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: 5. Without: 16. 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	ADRF	party	Everything lost by definition	The ADRF is the party that creates, notifies and deletes the subscription; with it controlled, every property asked here is lost by definition and the documents ask none of them under a dishonest ADRF (TS 33.501 5.9.1 assumes NFs of one operator lie in one trust zone, ts-33-501:2859).
S03	NRF	party	Everything lost by definition	The NRF is the authorization server; if the attacker controls it, it mints a token for any subject and the token properties are lost by definition.
S04	notifCorrId	value	`malicious_nf`	
S05	notificationURI	value	Nothing gained	A notificationURI is the callback address the ADRF must post to; it is a reachable network address and not a secret, so an attacker knowing it gains exactly the ability to send messages there.
S06	subscriptionId	value	`malicious_nf`	
S07	Subscription content	value	Nothing gained	The requested analytics subscription, data set and time window are not protected by any of the targets; only the notified payload is (K02).
S08	Notification instance	value	`malicious_nf`	
S09	Notification payload	value	Everything lost by definition	The payload is exactly the value K02 protects; an attacker holding it means K02 is lost by definition, so it cannot be an attacker for the properties asked here.
S10	timeStamp	value	Nothing gained	The time stamp of a notification is part of the payload that K02 covers; no target asks it to be secret on its own.
S11	OAuth2 token	value	`malicious_nf`	
S12	scope	value	Nothing gained	The scope name is public and is carried in the token; knowing it grants nothing the attacker does not already have.
S13	Audience	value	Nothing gained	The audience of a token is the ADRF's public identity.
S14	token audience	value	Nothing gained	The audience of a token is the ADRF's public identity.
S15	Subscription state	value	Nothing gained	Reading the subscription state gives an attacker nothing beyond the operations it can invoke; S05 asks about the notifications, not about secrecy of the state.
S16	anaNotifications	value	Everything lost by definition	The analytics notifications are the protected payload of K02; an attacker holding them means that property is lost by definition.
S17	dataNotif	value	Everything lost by definition	The data notification is the protected payload of K02; an attacker holding it means that property is lost by definition.

ID	Party or value	Kind	Attacker	Why none
S18	data/analytics retrieved by fetchInstruct	value	Outside the scope	Fetching the data with the fetch instruction uses the RetrievalRequest operation, which the task leaves out of the analysed sub-procedure.
S19	dataStoreId	value	Nothing gained	The data set identifier names stored records but grants no access in this API; the payload protection is K02.
S20	Subscription	value	Everything lost by definition	A subscription held by the attacker is the subscription whose properties A03 and Z03 ask about; nothing is left to ask.
S21	Data/analytics records	value	Outside the scope	The stored analytics records themselves are read with the RetrievalRequest operation, outside the analysed sub-procedure.

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" is 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`	The long-term credentials of one legitimate NF service consumer (its NF identity and its own OAuth2 access token with the scope "nadrf-datamanagement:storage-read-delete-subs"), from the start of the run; it can invoke every Nadrf_DataManagement service operation with that token and chooses the notifCorrId, the data set and the notificationURI of the subscriptions it creates. It also knows the callback address of another NF (a reachable network address) and the notifCorrId and subscriptionId that other NF uses, because those are strings for which the documents require uniqueness but never unguessability.	The documents allow any NF service consumer holding the API scope to invoke the RetrievalSubscribe, RetrievalUnsubscribe and RetrievalNotify operations (s0372-s0377); TS 29.501 4.10 states that a service-level-only scope grants all operations of the API to all consumers allowed to access the service (ts-29-501:1801). RFC 6749 10.2 states that a malicious client holding its own credentials can act, and TS 33.501 5.9.2.1 puts no restriction on which consumer may address which resource (ts-33-501:2871). The correlation identifier and the subscription identifier are strings for which the documents require uniqueness at most, never unguessability: notifCorrId 'shall be unique per subscription for a given NF service consumer' (s0325) and subscriptionId only 'identifies a subscription' (s0200).	ts-29-501:1801, ts-33-501:2871, s0325, s0200, s0372, s0375, s0377

ID	`network`	`malicious_nf`
A03	Holds	Holds
A04	Holds	Holds
D01	Holds	Fails
A08	Holds	Fails
Z01	Holds	Holds
Z03	Holds	Fails
S05	No answer	
R02	No answer	No answer
K02	Holds	Holds

ID	network	malicious_nf
K01	Holds	Holds
U01	Holds	Fails
U02	Holds	Holds
Z05	Holds	Fails

Source and targets

Source: 447 units, 331824 bytes, frozen in source-delivery-01.json. Targets: 12 approved (A03, A04, D01, A08, Z01, Z03, S05, R02, K02, K01, 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 the source package protocol-beba40bbab10b6aa5ff4a5b20e6f

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 152803 tokens in and 5151 out, 202 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
40	A03, A04, D01, A08, Z01, Z03, U01, U02, S05, R02, K02, K01, Z05	plain_primitives, many_sessions, other	The model uses only signatures, public-key encryption and hash-free tuples, with no algebraic law beyond that, and it covers any number of sessions because the ADRF, the NRF and the notification handler are replicated. The transport is represented as a signature by the sender and encryption to the receiver instead of TLS sessions, and the subscriptionId space is a single value, so the answer for A04 rests on a simplified identifier space. The subscription lifetime (removal then no further notification, S05 and the second half of U02) is represented by a monotonic table plus a removal marker rather than a real state machine, so the ordering claim there is not settled by this model: that needs a bounded state model (Tamarin+ Agent).

The analyst's closing note

The plain run (subscribe → 201 with subscriptionId → notification → 204 → DELETE → 204) was modelled first and kept throughout; the final model adds the NRF token, a second NF whose credentials the attacker holds, and a table of subscriptions. Twelve of the thirteen targets have an answer under their attackers, most of them proofs: the consumer's 201 matches the values it sent, the ADRF accepts only NRF-authorized NFs, every accepted notification is authentic and unmodified, the notification payload reaches no other NF, and no operation is accepted with a token the NRF did not issue.

Two weaknesses are established by trace. First, the notification carries only a notifCorrId that has to be unique per consumer, and the ADRF posts it to the notificationURI the subscriber names, so an NF that names another NF's callback address and its correlation value has its notifications accepted there, attributed to the victim's subscription and about a data set the victim never asked for (D01, A08, U01). Second, the DELETE carries nothing but the subscriptionId and no document requires the ADRF to compare the requester with the creator, so any NF holding the retrieval scope can remove another NF's subscription once it can name that identifier (Z03, Z05).

What stays open: R02 got no answer from ProVerif (inconclusive); S05 and the ordering half of U02 rest on a monotonic removal marker and A04's injectivity on a single-value subscriptionId; the transport is a signature-plus-encryption idealisation of mutual TLS. A bounded state model and Tamarin+ Agent are needed for the lifetime and identifier-space questions. Both recommendations (an ownership check on the DELETE, the subscriptionId in the notification) are given unchecked.