Security analysis of NWDAF training data management

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	13: 11 hold, 2 fail (2 attacks)
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 Nnwdaf_DataManagement service of an NWDAF lets another network function subscribe to aggregated non-personal training data, be notified when such data is ready, fetch it, and unsubscribe. Two parties take part: the NF service consumer, which holds its own OAuth2 access token with the single service scope nnwdaf-datamanagement and its own TLS session, and the NWDAF, which holds the subscription records and the collected data. The NRF acts as authorization server, and every request, notification and fetch travels over a mutually authenticated TLS connection of the service-based interface.

The exchange runs in four moves: the consumer posts a subscription carrying its notification URI and a correlation identifier, the NWDAF answers with the created subscription and its subscription identifier, notifies the consumer at that URI with fetch instructions, the consumer fetches the data with the correlation identifiers it was given, and it deletes the subscription. At the end the consumer is sure that each notification and payload carries the correlators of its own subscription and came over its own TLS connection, and the NWDAF is sure that whoever it served presented a token of the service scope and named a subscription record it holds.

The parties:

- **NF service consumer**: subscribes, is notified, fetches, unsubscribes. Holds: access token (scope nnwdaf-datamanagement), notifCorrId, notificURI.
- **NWDAF**: service producer. Holds: subscription records, collected and summarized data.
- **NRF**: issues the access token. Holds: NF profiles, acts as authorization server.

Figure 1 shows the exchange as the documents specify it.

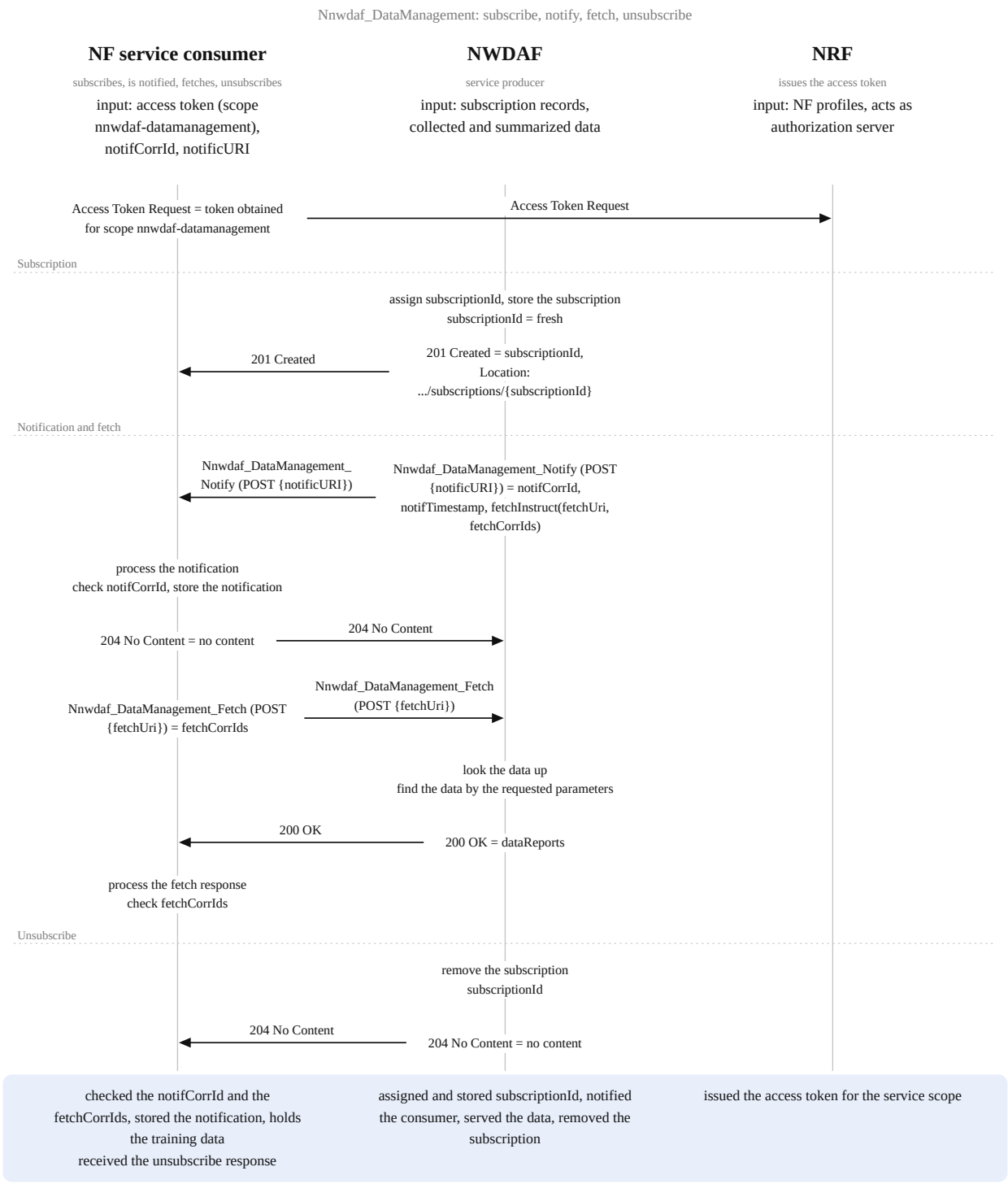


Figure 1. The protocol as specified.

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.

One finding: the API has a single service-wide scope and no check that the caller created the subscription, so a legal NF that knows a subscriptionId reads another consumer's training data by Fetch and deletes that consumer's subscription. Of the thirteen properties, six are proved and two are answered by that attack, while five are proved only in a weaker reading than they ask: A02, A03, A04, D01, A08 and U01 hold because the model keeps the notification endpoint and correlators from other NFs, which the specification never requires, and Z04, S05 and U03 are not settled because the model never removes the subscription record. The results rest on a symbolic model of TLS with atomic tokens and no token algebra, so they say nothing about computational strength.

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 A legal NF that knows a subscriptionId reads another consumer's training data and deletes its 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, U02 (section 3)
Source text	s0078, s0079, s0115, s0116, s0117, s0156, s0226, s0230, s0341, s0350
Shown by	ProVerif attack queries `z03`, `u02`; Figures 2 and 3

What happens. The NWDAF keys a subscription by its subscriptionId alone: a DELETE or a Fetch naming a subscription is served to any NF that presents a token of the single service scope. The creator of the subscription is never compared with the caller, and the application-error table has no rejection for a foreign resource. In the model the malicious NF deletes the honest consumer's subscription and is served its training data by Fetch.

What the attacker needs. A legal NF of the same network: its own access token for the single service scope, its own TLS connection to the NWDAF, and the victim's subscriptionId, which is a plain string in the resource URI and is declared confidential nowhere.

What the attacker gains. It reads the aggregated training data collected for that subscription and deletes the subscription, so it learns the data and denies the victim its notification and its fetched data.

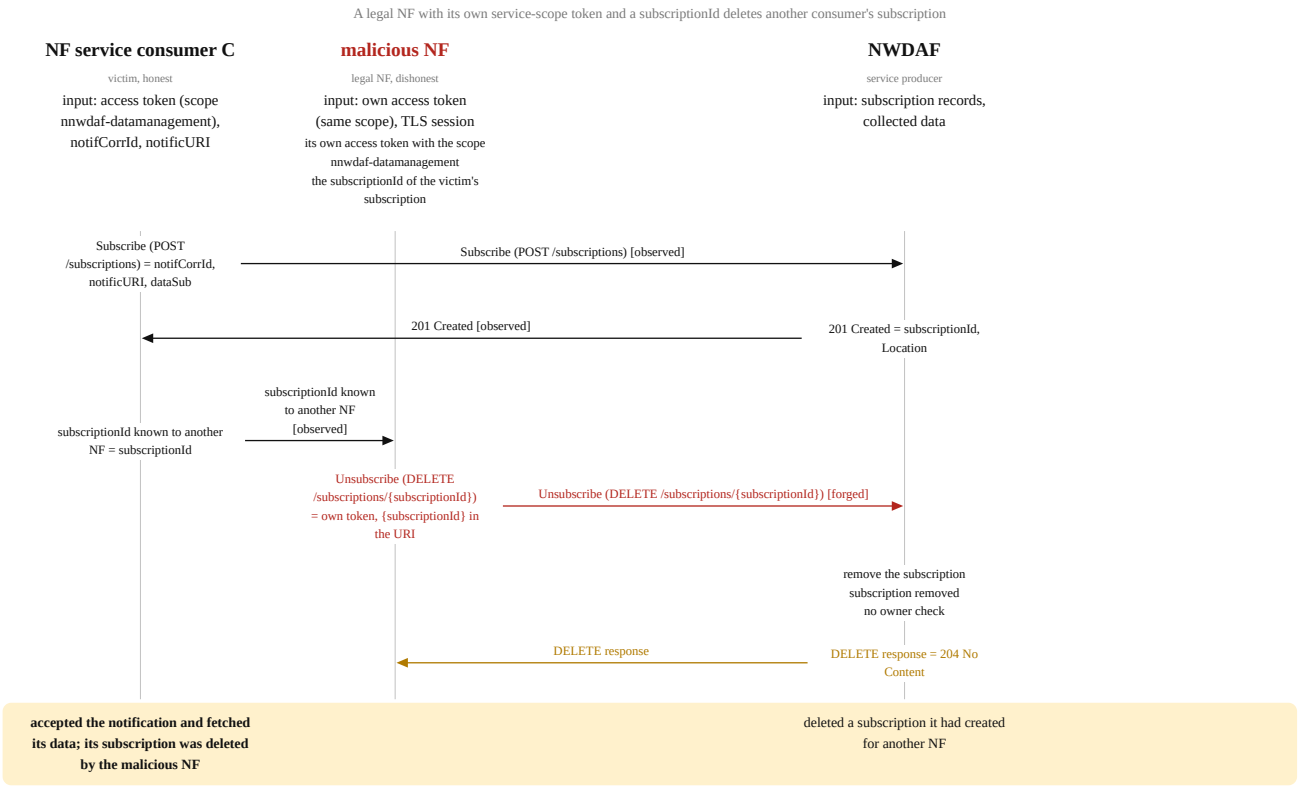


Figure 2. A legal NF with its own service-scope token and a subscriptionId deletes another consumer's subscription. Property Z03; attack query z03.

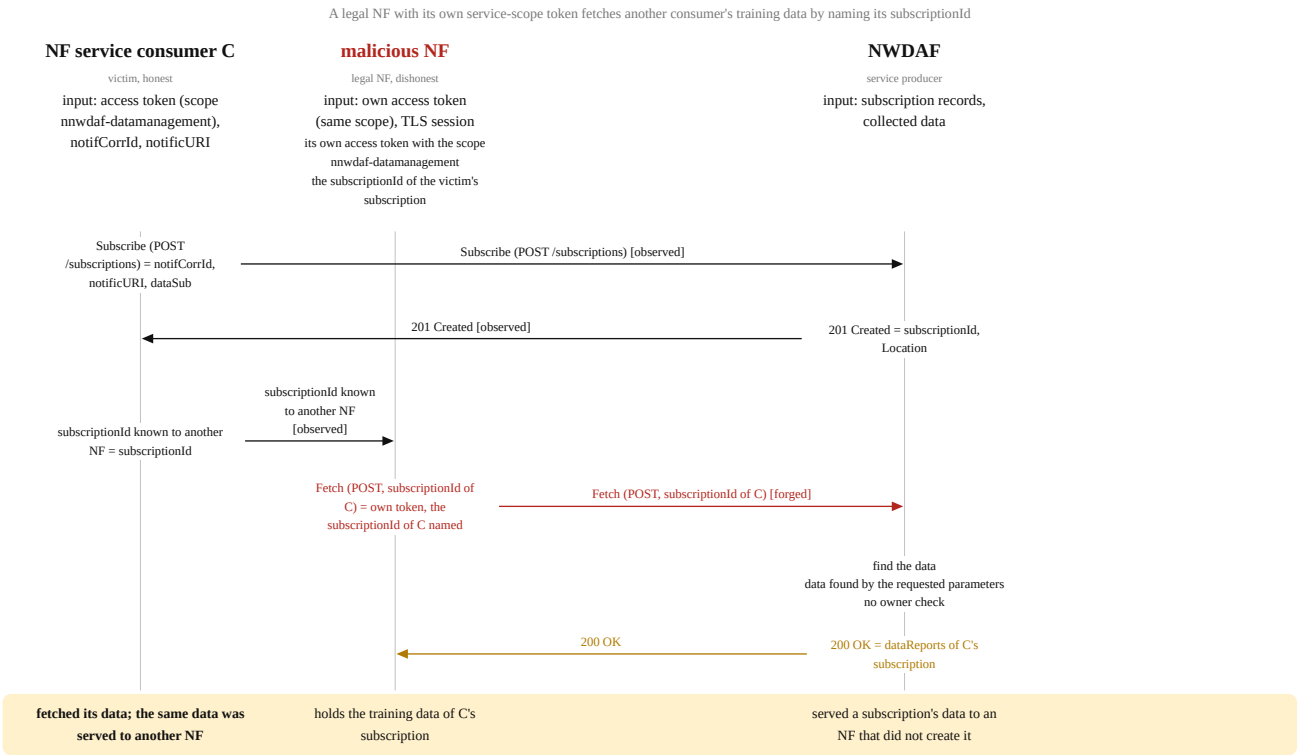


Figure 3. A legal NF with its own service-scope token fetches another consumer's training data by naming its subscriptionId. Property U02; attack query u02.

Proposed change. Record with each subscription the identity the access token was issued to, and compare it with the token identity of every later PUT, DELETE and Fetch on that subscriptionId, answering 403 Forbidden when they differ; a per-subscription scope would make the same check enforceable at the authorization server. This change was checked on a copy of the model: both attacks become unprovable, the normal run is still found, and of the other recorded answers only the two network-only readings change, because in the fixed model a legal NF is served its own subscription without anyone leaking a foreign identifier.

Check of the change with ProVerif. The change was put into a copy of the model: The NWDAF stores, with each subscription, the identity under which the subscription was created (the identity the Nnwdaf_DataManagement_Subscribe request was authorized for) and compares it with the identity of the requester of every later Fetch and DELETE on that subscriptionId, answering 403 Forbidden when they differ instead of serving the deletion or the data. On the changed model ProVerif proves `z03`: the attack is no longer possible there. The normal run of the protocol is still found. Of the other recorded answers, 23 still hold and 3 changed: `z03_net` (Z03, was proved) is now falsified; `u02_net` (U02, was proved) is now falsified; `u02` (U02, was attack) is now verified. 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/0040-fix-owner.pv`.

No figure was given for these attacks of a finding: U02: z03, Z03: u02.

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
A02	**Notification sender is the NWDAF.** Whenever a consumer accepts a notification or subscription response from the NWDAF, there exists a real NWDAF run whose intended identity matches the peer identity the consumer believes it is talking to.	no compromise	Holds	low	no
A03	**Correlators bind notification to subscription.** Whenever a consumer accepts a notification or a Fetch response, there exists a matching NWDAF run whose role, both identities and correlation fields such as notifCorrId, fetchCorrIds and subscriptionId agree with the consumer's local values.	no compromise	Holds	low	no
A04	**One acceptance per NWDAF run.** Each time a consumer accepts a subscription response or a notification, it corresponds to a distinct NWDAF run; no two acceptances map to the same NWDAF run.	no compromise	Holds	low	no
D01	**Notification payload comes from the NWDAF.** Each time a consumer accepts notification data, there exists a real send by the permitted sender NWDAF, and the payload agrees with the approved correlation fields.	no compromise	Holds	low	no
A08	**Notification fields match the subscription.** For the notifications and Fetch responses a consumer accepts, the fields such as notificURI, notifCorrId, fetchUri and fetchCorrIds correspond to the values the consumer proposed in the subscription and belong to the current security context.	no compromise	Holds	low	no
K01	**Access token and SBI keys secret.** An attacker cannot derive the consumer's OAuth2 access token or the SBI session key between the NWDAF and the consumer.	no compromise	Holds	low	yes

ID	Property	Compromise assumed	Result	Value	In the documents
Z01	**Every operation is authorized.** Each time the NWDAF submits a subscription create/update/delete or returns Fetch data, there exists a grant issued by the authorization server that is valid for the requester and covers the nnwdaf-datamanagement scope.	no compromise	Holds	medium	yes
Z03	**Operation subject is subscription creator.** For each update, delete or Fetch on a given subscriptionId, the authorization subject is the same as the creator of that subscription, and the operation matches the token audience and resource.	no compromise	Fails (1 attack)	medium	unclear
Z04	**No notification after termination.** After a subscription is deleted or the NWDAF request is terminated, no new notifications or Fetch data are produced for that subscriptionId.	no compromise	Holds	none	no
S05	**Old identifiers stop working.** After the point at which subscription termination takes effect, using the old subscriptionId, notifCorrId or fetchUri cannot trigger new notification delivery or Fetch data return.	no compromise	Holds	none	no
U01	**Correlators only for their consumer.** The notifCorrId/fetchCorrIds in notifications and Fetch are valid only for the consumer that initiated the subscription.	no compromise	Holds	low	no
U02	**Other NF cannot read or replace data.** A malicious legal NF cannot replace or read the data of other subscriptions.	no compromise	Fails (1 attack)	medium	unclear
U03	**No notifications after unsubscribe.** After the unsubscribe takes effect, the consumer accepts no further notifications for that subscription.	no compromise	Holds	none	no

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: The correId of subscriptions/notifications/Fetch is bound to the requester	Only the consumer side is bound: a consumer accepts a notification or a fetch response only with the correlator it chose itself, and this is verified. On the producer side neither correlator is bound to a consumer identity, so a correlation identifier is not an authorization token: another legal NF that knows the subscriptionId is served under its own correlators.	U01, A08, Z03, U02
Q2: Cross-subscription data replacement and the read boundary for a malicious legal NF	A malicious legal NF with its own service-scope token reads another consumer's training data by invoking Fetch with a subscriptionId it did not create and deletes that subscription; both attacks hold on the model, so the boundary the API gives is the secrecy of the resource identifier, not the token. It cannot inject notifications into the victim's consumer on this model, because the victim's notification endpoint and correlator are unknown to it.	U02, Z03, A02, A03
Q3: Limited number of notifications and handling of stale responses after cancellation	Neither is settled: the specification sets no limit on the number of notifications, and the model cannot show that nothing arrives after the unsubscribe because the subscription record is never removed in it. Both stay open and need a stateful model of the record.	Z04, S05, U03

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 modelled as a private channel per endpoint: the network attacker reads no plaintext and cannot forge or replay, and a peer's certificate identity is never compared with the identity it is expected to have.
- The notification endpoint and the correlation identifiers are treated as known only to the NWDAF that uses them; a legal NF that learns another consumer's notificURI and notifCorrId is not modelled, and the specification requires no such secrecy.
- The subscription record is never removed from the store of the model, so no claim about what follows termination is settled in the sense the targets ask; Z04, S05 and U03 are answered by weaker readings only.
- The OAuth2 token is an atomic value that is either accepted or not: audience, expiry and the scope-to-resource semantics of TS 29.510 and TS 33.501 are not modelled, and no token is stolen or replayed.
- Application errors, muting, storage handling, pending notifications and redirects are not modelled; only the plain four-exchange run and a legal NF's own requests are.
- UE consent, personal data, ADRF, DCCF, MFAF, multi-source collection and the processing function are outside the scope the task sets.
- Symbolic model: no computational claim, no algorithm strength, no quantum resistance.

What the host records:

- Proved without a control query (Appendix C): A02: a02, A03: a03, A04: a04, D01: d01, A08: a08, K01: k01, Z01: z01, Z01: z01b, Z04: term_notif, S05: term_data, U01: u01, U03: term_notif. 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.

A02: Notification sender is the NWDAF

Whenever a consumer accepts a notification or subscription response from the NWDAF, there exists a real NWDAF run whose intended identity matches the peer identity the consumer believes it is talking to.

Every notification the consumer accepts came from the NWDAF for that subscription with the correlator the consumer chose. The consumer checks the correlator only; the sender's identity comes from TLS, and no comparison with the NWDAF it subscribed to is required.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It follows from the modelling of the endpoint, not from a stated rule.
- Stated in the documents: no
- Rests on: The notification endpoint and correlator are modelled as unknown to other NFs, which the specification does not require.
- Open: A legal NF that learns another consumer's notificURI is not covered.

A03: Correlators bind notification to subscription

Whenever a consumer accepts a notification or a Fetch response, there exists a matching NWDAF run whose role, both identities and correlation fields such as notifCorrId, fetchCorrIds and subscriptionId agree with the consumer's local values.

An accepted fetch response carries exactly the correlators and payload the NWDAF produced for that subscription, and the consumer checks them before accepting. Correlation is sound on the consumer side, which does not make the identifiers authorization data.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: A consequence of the endpoint modelling rather than of a rule of the specification.
- Stated in the documents: no
- Rests on: The fetch endpoint is a channel the NWDAF created and handed out in the notification.
- Open: A legal NF that learns the fetchCorrIds is not modelled.

A04: One acceptance per NWDAF run

Each time a consumer accepts a subscription response or a notification, it corresponds to a distinct NWDAF run; no two acceptances map to the same NWDAF run.

Each acceptance of a subscription response matches a distinct NWDAF send, so a replayed message cannot produce a second acceptance. This comes from the fresh connection per exchange, not from a freshness value in the messages.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: Injectivity comes from the private-channel model.

- Stated in the documents: no
- Rests on: The transport is a linear private channel; the specification states no replay protection for these messages.
- Open: HTTP-level replay is not modelled.

D01: Notification payload comes from the NWDAF

Each time a consumer accepts notification data, there exists a real send by the permitted sender NWDAF, and the payload agrees with the approved correlation fields.

The data a consumer accepts by fetch is the data the NWDAF produced for that subscription, and nothing else can be substituted on the path. The protection is the transport alone.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It restates the transport assumption.
- Stated in the documents: no
- Rests on: SBI protection modelled as a private channel (s0347).
- Open: Integrity of the notification body beyond the transport is not modelled.

A08: Notification fields match the subscription

For the notifications and Fetch responses a consumer accepts, the fields such as notificURI, notifCorrId, fetchUri and fetchCorrIds correspond to the values the consumer proposed in the subscription and belong to the current security context.

The correlator a consumer sees is always one it proposed, and the notification arrives on the endpoint it created for that subscription, so the fields of an accepted notification belong to its own subscription.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It follows from how the endpoints are modelled.
- Stated in the documents: no
- Rests on: The notification endpoint is a channel the consumer created and passed in the subscribe request.
- Open: Delivery to a wrong endpoint is outside the model.

K01: Access token and SBI keys secret

The attacker cannot derive the consumer's OAuth2 access token or the SBI session keys between the NWDAF and the consumer.

The consumer's token and the session keys of its connection never reach the attacker, and nothing in the exchange derives them. The arrangement is the OAuth2 client-credentials one, and the API flow adds no path to the token.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The result is the assumption the specification already makes.
- Stated in the documents: yes. Where the documents say it: rfc6749:2859, ts-33-501:2885
- Rests on: The token is an atomic value; TLS keys are represented only by the connection being private.
- Open: Token theft, replay and expiry are not modelled.

Z01: Every operation is authorized

Every time the NWDAF submits a subscription create/update/delete or returns Fetch data, there is a permit issued by the authorization server that is valid for the requester and covers the nnwdaf-datamanagement scope.

Every delete and every fetch served was preceded by a token the NWDAF accepts for the service scope, so nothing is served unauthorized. Because one scope covers the whole service, an accepted token shows the caller is some authorized NF and nothing about which records it may touch.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It fixes the exact sense in which the service is authorized, the premise on which the Z03 and U02 failures rest; the fact itself is in the documents.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2885, main:3460
- Rests on: Token validation is an atomic check; audience, expiry and token-to-identity binding are outside the model.
- Open: Whether the scope ought to cover a foreign resource is exactly what Z03 and U02 answer.

Z03: Operation subject is subscription creator

Every update, delete or Fetch on a subscriptionId has an authorization subject that matches the creator of that subscription, and the operation matches the token audience and the resource.

A DELETE is served to any NF that presents a token of the service scope and knows the subscriptionId; the creator is never compared with the caller. Against the network alone no deletion is reachable, so the weakness is exactly as wide as the resource identifier is knowable.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: A service-wide scope with no resource check lets a legal NF delete a colleague's subscription; the documents state the scope but not this consequence.
- Stated in the documents: unclear. Source text it rests on: s0115, s0116, s0117, s0226, s0230, s0341, s0350
- Rests on: The subscriptionId is treated as obtainable: a plain string in the URI, no confidentiality declared, no resource-level scope.
- Open: The attacker's knowledge of a foreign subscriptionId is a stated capability, not derived from a leak inside the protocol.; An implementation adding an ownership check on top of the scope is not modelled.

Z04: No notification after termination

After the subscription is deleted or the NWDAF requests termination, no new notification or Fetch data is produced for that subscriptionId.

The recorded answer shows only that a notification accepted around the termination is a real NWDAF notification for that subscription. Whether the NWDAF could still notify after it answered the deletion is not established.

- Result: Holds; answers what is asked: partly; not usable
- Value none: The answer speaks about the notification path, not about what happens after termination, which is what the target asks.
- Stated in the documents: no. Source text it rests on: s0116, s0117, s0119, s0120, s0140
- Rests on: Only the plain run is unrolled; the subscription record is never emptied.
- Open: What follows the termination is not settled; the record is never removed in the model.; The 200 OK answer with unsent stored events is not modelled.

S05: Old identifiers stop working

After the point at which subscription termination takes effect, using the old subscriptionId, notifCorrId or fetchUri cannot trigger new notification delivery or Fetch data returns.

The recorded answer shows that the payload the consumer accepts around its unsubscribe is the NWDAF's. That an old subscriptionId or fetchUri no longer yields data cannot be shown, because the model never removes the record it serves from.

- Result: Holds; answers what is asked: partly; not usable
- Value none: It answers a neighbouring claim, not the one the target makes.
- Stated in the documents: no. Source text it rests on: s0116, s0117, s0140
- Rests on: The stored subscription is never deleted from the prover's table.
- Open: Whether an old identifier stops working after deletion is not settled.; Invalidation of the endpoints and correlators is not modelled.

U01: Correlators only for their consumer

The notifCorrId/fetchCorrIds in notifications and Fetch are valid only for the consumer that initiated that subscription.

A correlation identifier is meaningful only in the consumer that chose it: no other consumer's endpoint accepts it and the NWDAF cannot attribute it to a different consumer. It is a label, not a credential, which is why it protects the consumer but not the producer.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It follows from the endpoint modelling; the specification states no binding rule.
- Stated in the documents: no
- Rests on: Each consumer's notification endpoint is known only to the NWDAF of that subscription.
- Open: A consumer deliberately reusing another consumer's correlator is not modelled.

U02: Other NF cannot read or replace data

A malicious legal NF cannot replace or read the data of other subscriptions.

A legal NF holding a service-scope token and another consumer's subscriptionId is served that subscription's training data by Fetch, with no check that it created the subscription. Against the network alone nothing is served, so the secrecy of the resource identifier is the whole protection of the data.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The missing resource-level authorization gives a legal NF read access to another consumer's training data; the documents state the scope design, not the loss.
- Stated in the documents: unclear. Source text it rests on: s0156, s0157, s0158, s0159, s0230, s0341, s0350
- Rests on: The subscriptionId is treated as obtainable (s0226, s0230, s0350).; The NWDAF serves any fetch whose subscriptionId it holds, as 4.4.2.5.2 requires.
- Open: How a legal NF obtains a foreign subscriptionId is assumed, not derived.; Whether this data is sensitive in an operator's policy is a deployment matter.

U03: No notifications after unsubscribe

After the unsubscribe takes effect, the consumer no longer accepts notifications for that subscription.

In the plain run the consumer accepts the notification, fetches, deletes and ends, so nothing follows the unsubscribe; the recorded answer adds only that a notification around that moment is still the NWDAF's. Whether an in-flight notification could still be accepted afterwards is not answered.

- Result: Holds; answers what is asked: partly; not usable
- Value none: It does not establish the claim the target makes.
- Stated in the documents: no. Source text it rests on: s0116, s0117, s0119, s0120
- Rests on: One notification per subscription, linear consumer flow.
- Open: That no notification is accepted after the unsubscribe needs an ordering of the deletion against delivery, which the model does not build.

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
A02	proved		a02			2 proved
A03	proved		a03			2 proved
A04	proved		a04			2 proved
D01	proved		d01			2 proved
A08	proved		a08			2 proved
K01	proved		k01			2 proved
Z01	proved		z01			4 proved
Z03	proved		z03_net	checked	z03 (holds_in_source)	1 proved, 1 attack
Z04	proved		term_notif			2 proved
S05	proved		term_data			2 proved
U01	proved		u01			2 proved
U02	proved		u02_net	checked	u02 (holds_in_source)	1 proved, 1 attack
U03	proved		term_notif			2 proved

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
A02	Whenever the consumer accepts a notification, a real NWDAF run sent it, for the same subscription and the same notifCorrId.	`network`	proved	a02	none
A02	Whenever the consumer accepts a notification, a real NWDAF run sent it, for the same subscription and the same notifCorrId, even when another legal NF knows the consumer's subscriptionId.	`malicious_nf_resource_id`	proved	a02	none
A03	An accepted fetch response carries exactly the fetchCorrIds and the payload the NWDAF sent for that subscription.	`network`	proved	a03	none
A03	An accepted fetch response carries exactly the fetchCorrIds and the payload the NWDAF sent for that subscription, even when another legal NF knows the consumer's subscriptionId.	`malicious_nf_resource_id`	proved	a03	none
A04	Every acceptance of a subscription response is matched by a distinct NWDAF send of that subscription response; replay of one NWDAF send cannot produce two acceptances.	`network`	proved	a04	none

target	claim	attacker	answer	query	control
A04	Every acceptance of a subscription response is matched by a distinct NWDAF send of that subscription response, even when another legal NF knows the consumer's subscriptionId.	`malicious_nf_resource_id`	proved	a04	none
D01	The dataNotification/dataReports a consumer accepts by fetch were sent by the NWDAF for that subscription.	`network`	proved	d01	none
D01	The dataNotification/dataReports a consumer accepts by fetch were sent by the NWDAF for that subscription, even when another legal NF knows the consumer's subscriptionId.	`malicious_nf_resource_id`	proved	d01	none
A08	The notifCorrId of an accepted notification is one the consumer itself proposed in the subscribe request for that subscription.	`network`	proved	a08	none
A08	The notifCorrId of an accepted notification is one the consumer itself proposed in the subscribe request for that subscription, even when another legal NF knows the consumer's subscriptionId.	`malicious_nf_resource_id`	proved	a08	none
K01	The consumer's OAuth2 access token and the SBI session keys between consumer and NWDAF cannot be derived by the attacker.	`network`	proved	k01	none
K01	The consumer's OAuth2 access token and the SBI session keys between consumer and NWDAF stay secret from a legal NF that holds its own token.	`malicious_nf_resource_id`	proved	k01	none
Z01	A subscription deletion is served only after the request presented a token the NWDAF accepts for the nnwdaf-datamanagement service scope.	`network`	proved	z01	none
Z01	A subscription deletion is served only after the request presented a token the NWDAF accepts for the nnwdaf-datamanagement service scope, also when the requester is a legal NF holding its own token.	`malicious_nf_resource_id`	proved	z01	none
Z01	A fetch response is returned only after the request presented a token the NWDAF accepts for the nnwdaf-datamanagement service scope.	`network`	proved	z01b	none
Z01	A fetch response is returned only after the request presented a token the NWDAF accepts for the nnwdaf-datamanagement service scope, also when the requester is a legal NF holding its own token.	`malicious_nf_resource_id`	proved	z01b	none
Z03	A subscription deletion is served only to the NF that created that subscription; with the network alone the attacker cannot reach any subscription.	`network`	proved	z03_net	checked
Z03	A subscription deletion is served only to the NF that created that subscription; a legal NF holding its own service-scope token cannot delete another consumer's subscription.	`malicious_nf_resource_id`	attack (holds_in_source)	z03 (the control of z03_net)	

target	claim	attacker	answer	query	control
Z04	A notification the consumer accepts for a subscription that is being terminated is still one the NWDAF sent for that subscription; no notification for a terminated subscription reaches the consumer from elsewhere.	`network`	proved	term_notif	none
Z04	A notification the consumer accepts for a subscription that is being terminated is still one the NWDAF sent for that subscription, even when another legal NF knows the subscriptionId.	`malicious_nf_resource_id`	proved	term_notif	none
S05	A fetch payload the consumer accepts for a subscription that is being terminated is still the payload the NWDAF sent for that subscription.	`network`	proved	term_data	none
S05	A fetch payload the consumer accepts for a subscription that is being terminated is still the payload the NWDAF sent for that subscription, even when another legal NF knows the subscriptionId.	`malicious_nf_resource_id`	proved	term_data	none
U01	A notifCorrId the consumer accepts on a notification is one it chose itself; no other consumer can have it accepted.	`network`	proved	u01	none
U01	A notifCorrId the consumer accepts on a notification is one it chose itself; no other consumer, even one knowing the subscriptionId, can have it accepted.	`malicious_nf_resource_id`	proved	u01	none
U02	With the network alone no other NF reads the data of a subscription: every read the NWDAF serves to another NF requires that NF to know the subscriptionId.	`network`	proved	u02_net	checked
U02	A malicious legal NF holding its own token of the service scope cannot read or replace another NF's training data.	`malicious_nf_resource_id`	attack (holds_in_source)	u02 (the control of u02_net)	
U03	A notification the consumer accepts around its unsubscribe is still one the NWDAF sent for that subscription, so the unsubscribe response is not followed by notifications from elsewhere.	`network`	proved	term_notif	none
U03	A notification the consumer accepts around its unsubscribe is still one the NWDAF sent for that subscription, even when another legal NF knows the subscriptionId.	`malicious_nf_resource_id`	proved	term_notif	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-20261114T213347, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 43, prover 4 s, wall 11 min, cost 0.097 USD.

ProVerif: 87 proofs run, 87 decided, 0 without an answer, 3 s, 39 answers recorded.

What was analysed

Protocol	NWDAF training data management
Version	19.8.0
Analysed	Two NWDAFs directly Subscribe/Notify/Fetch/Unsubscribe; fixed aggregated non-personal training data, a single subscription, the data processing function is opaque; ADRF, DCCF, multi-source forwarding, and the UE consent branch are excluded.
Not analysed	The symbolic model does not prove computational or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Each property must have a reachability/positive control. Rel-19 is fixed. The external network attacker and the malicious legal NF holding its own credentials are distinguished separately; TLS authenticated encryption is modelled per endpoint, and a legal 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. Monotonic tables cannot model real deletion, and the finite-state lifecycle must be explicitly defined. The target is the specified API sub-flow, not complete 5G security.
Source text	350 numbered passages (73904 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	4.4	Nnwdaf_DataManagement Service
main	4.4.1	Service Description
main	4.4.1.1	Overview
main	4.4.1.2	Service Architecture
main	4.4.1.3	Network Functions
main	4.4.1.3.1	Network Data Analytics Function (NWDAF)
main	4.4.1.3.2	NF Service Consumers
main	4.4.2	Service Operations
main	4.4.2.1	Introduction
main	4.4.2.2	Nnwdaf_DataManagement_Subscribe service operation
main	4.4.2.2.1	General
main	4.4.2.2.2	Subscription for data notifications
main	4.4.2.2.3	Update subscription for data notifications

Document	Section	Title
main	4.4.2.3	Nnwdaf_DataManagement_Unsubscribe service operation
main	4.4.2.3.1	General
main	4.4.2.3.2	Unsubscribe from data notifications
main	4.4.2.4	Nnwdaf_DataManagement_Notify service operation
main	4.4.2.4.1	General
main	4.4.2.4.2	Notification about subscribed data
main	4.4.2.5	Nnwdaf_DataManagement_Fetch service operation
main	4.4.2.5.1	General
main	4.4.2.5.2	Retrieve data from the NWDAF
main	5.3	Nnwdaf_DataManagement Service API
main	5.3.1	Introduction
main	5.3.2	Usage of HTTP
main	5.3.2.1	General
main	5.3.2.2	HTTP standard headers
main	5.3.2.2.1	General
main	5.3.2.2.2	Content type
main	5.3.2.3	HTTP custom headers
main	5.3.3	Resources
main	5.3.3.1	Resource Structure
main	5.3.3.2	Resource: NWDAF Data Management Subscriptions
main	5.3.3.2.1	Description
main	5.3.3.2.2	Resource Definition
main	5.3.3.2.3	Resource Standard Methods
main	5.3.3.2.3.1	POST
main	5.3.3.2.4	Resource Custom Operations
main	5.3.3.3	Resource: Individual NWDAF Data Management Subscription
main	5.3.3.3.1	Description
main	5.3.3.3.2	Resource definition
main	5.3.3.3.3	Resource Standard Methods
main	5.3.3.3.3.1	PUT
main	5.3.3.3.3.2	DELETE
main	5.3.3.3.4	Resource Custom Operations
main	5.3.4	Custom Operations without associated resources
main	5.3.5	Notifications
main	5.3.5.1	General
main	5.3.5.2	Event Notification

Document	Section	Title
main	5.3.5.2.1	Description
main	5.3.5.2.2	Operation Definition
main	5.3.5.3	Fetch Notification
main	5.3.5.3.1	Description
main	5.3.5.3.2	Target URI
main	5.3.5.3.3	Standard Methods
main	5.3.5.3.3.1	POST
main	5.3.6	Data Model
main	5.3.6.1	General
main	5.3.6.2	Structured data types
main	5.3.6.2.1	Introduction
main	5.3.6.2.2	Type NnwdafDataManagementSubsc
main	5.3.6.2.3	Type NnwdafDataManagementNotif
main	5.3.6.3	Simple data types and enumerations
main	5.3.6.3.1	Introduction
main	5.3.6.3.2	Simple data types
main	5.3.6.3.3	Enumeration: PendingNotificationCause
main	5.3.7	Error handling
main	5.3.7.1	General
main	5.3.7.2	Protocol Errors
main	5.3.7.3	Application Errors
main	5.3.8	Feature negotiation
main	5.3.9	Security

The purpose of the protocol as the source step located it in the documents: two NWDAFs Subscribe/Notify/Fetch/Unsubscribe directly; fixed aggregation of non-personal training data, a single subscription, the data processing function is opaque, excluding ADRF, DCCF, multi-source forwarding, and the UE consent branch.

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
A02	Notification sender is the NWDAF	The NF service consumer is assured that a POST it accepts on its notificURI as a notification or subscription response comes from the NWDAF it subscribed to, not from another NF.	The consumer subscribes, is notified at notificURI and accepts and stores the notification (s0006-s0008, s0125, s0130, s0142); the only stated protection is SBI mutual authentication (s0347, ts-33-501:2871), which authenticates a peer NF, not the particular NWDAF. No sender check at the consumer beyond that is stated.	ts-33-501:2871, s0006, s0007, s0125, s0130, s0142, s0347
A03	Correlators bind notification to subscription	An accepted notification or fetch response carries the notifCorrId/fetchCorrIds and subscriptionId of the consumer's own subscription, not another consumer's.	The request carries notificURI and notifCorrId (s0053, s0054), the NWDAF assigns subscriptionId (s0079, s0081), the notification repeats notifCorrId (s0132) and fetch uses fetchCorrIds (s0137, s0156). No generation, uniqueness or consumer-binding rule is stated.	s0053, s0054, s0079, s0081, s0132, s0137, s0156
A04	One acceptance per NWDAF run	Two acceptances of a subscription response or notification do not map to one NWDAF run: replay does not produce a second acceptance.	A new subscriptionId is assigned per created subscription (s0079, s0081) and unsubscribe removes the subscription (s0116, s0117); notifTimestamp is only the preparation time (s0133). No replay protection is stated for these messages.	s0079, s0081, s0116, s0117, s0133
D01	Notification payload comes from the NWDAF	Every dataNotification/dataReports value the consumer accepts was sent by the NWDAF for that subscription and correlation identifier.	The NWDAF sends the notification with dataNotification/dataReports (s0125, s0130, s0135, s0136) and the consumer accepts it if processing succeeds (s0142); no message-level integrity beyond the SBI is stated (s0347).	s0125, s0130, s0135, s0136, s0142
A08	Notification fields match the subscription	The notification and fetch response the consumer accepts correspond to the notificURI, notifCorrId, fetchUri and fetchCorrIds it proposed, in the context of its own subscription.	The consumer proposes notificURI, notifCorrId, anaSub/dataSub and processing instructions (s0053-s0057, s0060, s0061); the NWDAF notifies to that URI with that notifCorrId (s0130-s0132) and hands out fetchUri/fetchCorrIds (s0137, s0156). No binding rule is stated.	s0053, s0054, s0056, s0057, s0060, s0061, s0130, s0132, s0137, s0156
K01	Access token and SBI keys secret	The consumer's OAuth2 access token and the SBI session keys between consumer and NWDAF are kept from the attacker.	Access may be authorized with OAuth2 client credentials obtained from the NRF (s0347, s0348) and the SBI is protected as TS 29.500/TS 33.501 require (s0347); token and key formats are outside the selected units.	rfc6749:2859, s0347, s0348
Z01	Every operation is authorized	Every subscription creation, deletion and fetch response is covered by a token issued by the authorization server, valid for the requesting NF, carrying the nnwdaf-datamanagement scope.	Access may be authorized by OAuth2 client credentials with the NRF as authorization server (s0347), a token is obtained before consuming the service (s0348), and one scope nnwdaf-datamanagement covers the whole service (s0350).	ts-33-501:2885, s0347, s0348, s0350
Z03	Operation subject is subscription creator	The subject of a DELETE on, or a Fetch under, a subscriptionId is the NF that created that subscription; otherwise the NWDAF rejects the operation.	The NWDAF creates the subscription and assigns subscriptionId (s0078, s0079); unsubscribe deletes by subscriptionId (s0115, s0117) and fetch is by fetchCorrIds (s0156). The single service scope (s0350) and the error table (s0341) name no ownership rule or rejection for a foreign resource.	s0078, s0079, s0115, s0117, s0156, s0341, s0350
Z04	No notification after termination	After the subscription is deleted, no further notification for that subscriptionId reaches the consumer.	On DELETE the NWDAF removes the subscription and answers 204 (s0116-s0119); a termination request can be indicated in a notification (s0140). Unsent stored events may be returned in the DELETE response (s0120); the timing of in-flight notifications is not specified.	s0116, s0117, s0119, s0120, s0140

ID	Name	About	What the documents say	Where
S05	Old identifiers stop working	After termination an old subscriptionId, notifCorrId or fetchUri no longer yields notification delivery or fetched data.	Deletion removes the subscription (s0116, s0117) and a termination request may be indicated (s0140); no rule on invalidating the identifiers or on cached data is stated.	s0116, s0117, s0140
U01	Correlators only for their consumer	A notifCorrId chosen by one consumer is meaningful only for that consumer: another consumer can neither make the NWDAF use it nor accept notifications carrying it.	notifCorrId is chosen by the consumer (s0054) and repeated by the NWDAF (s0132); fetchCorrIds are handed out by the NWDAF (s0156). No binding of these identifiers to the consumer identity is stated.	s0054, s0132, s0156
U02	Other NF cannot read or replace data	A malicious legal NF holding its own token of the service scope cannot read or replace the training data of another NF's subscription.	The service collects training data for the subscribing consumer and returns it by notification or fetch (s0006-s0008, s0125, s0159-s0161); the API has one service-wide scope (s0350) and names no error for a foreign resource (s0341).	ts-33-501:2859, s0006, s0007, s0008, s0125, s0159, s0161, s0341, s0350
U03	No notifications after unsubscribe	Once the consumer's unsubscribe has completed, the consumer accepts no further notification of that subscription.	On DELETE the NWDAF removes the subscription (s0116, s0117) and answers 204 or 200 with unsent stored events (s0119, s0120); no rule limits the number of notifications or orders them against the DELETE response.	s0116, s0117, s0119, s0120

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: 48. Not taken: 350. 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 the RFC 6749 security considerations: advice to implementers, no property of this API, no attacker.
P003 to P006	rfc6749:2790-2823	Z01, `network`	
P007 to P010	rfc6749:2827-2850	Z03, U02, `malicious_nf_resource_id`	
P011 to P015	rfc6749:2859-2881	K01, Z01, `network`	
P016 to P021	rfc6749:2885-2917	Outside the scope	Refresh tokens; the API uses the client-credentials grant, which issues none (s0347).
P022 to P025	rfc6749:2921-2943	Outside the scope	Authorization code grant; not used (s0347).
P026 to P029	rfc6749:2947-2980	Outside the scope	Redirect URI manipulation in the authorization code flow; not part of this grant.
P030 to P033	rfc6749:2984-3003	Outside the scope	Resource owner password credentials grant; not used (s0347).
P034 to P035	rfc6749:3013-3019	K01, `network`	

Paragraphs	Where	Taken up as	Why not
P036	rfc6749:3023-3028	A02, `network`	
P037 to P039	rfc6749:3032-3041	K01, Z01, `network`	
P040 to P042	rfc6749:3045-3066	Outside the scope	Phishing of resource owners; there is no resource owner in the client-credentials grant.
P043 to P048	rfc6749:3070-3110	Outside the scope	Cross-site request forgery of a browser user agent; no browser in this NF-to-NF sub-flow.
P049 to P050	rfc6749:3119-3138	Outside the scope	Clickjacking; browser framing, not applicable to SBI requests.
P051 to P052	rfc6749:3142-3150	A08	
P053 to P054	rfc6749:3154-3167	Outside the scope	Open redirectors: this API's redirect responses are SCP-level (s0122, s0148, s0163) and not modelled.
P055 to P062	rfc6749:3170-3210	Outside the scope	Access tokens misused in the implicit grant; this API uses client credentials and no user agent presents a token.
P063	ts-33-501:2491-2491	The prover cannot be asked	Bidding down a security feature: algorithm strength and feature negotiation (s0343) cannot be asked of a symbolic prover.
P064 to P072	ts-33-501:2495-2511	Outside the scope	Authentication and authorization of the UE and of the serving network by the home network; this sub-flow has no UE.
P073 to P125	ts-33-501:2515-2631	Outside the scope	Access-network and gNB keys, user-plane and signalling confidentiality and integrity, and subscriber privacy; the task leaves personal data, UE consent and radio keys out.
P126 to P196	ts-33-501:2637-2805	Outside the scope	Requirements on gNB, ng-eNB and their interfaces, a part of the system this task leaves out.
P197 to P213	ts-33-501:2809-2853	Outside the scope	Requirements on SEAF, UDM, AUSF and UDR in subscriber authentication; no subscription data request is made in this sub-flow (UE consent excluded).
P214	ts-33-501:2859-2859	U02, `malicious_nf_resource_id`	
P215 to P220	ts-33-501:2865-2875	Z01, A02, A03, `network`	
P221 to P227	ts-33-501:2879-2891	Z01, K01	
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF security requirements; the NEF is not a party of this sub-flow.
P235 to P241	ts-33-501:2911-2923	Outside the scope	Service Communication Proxy; the task models direct NF-to-NF communication without an SCP.
P242 to P254	ts-33-501:2929-2953	A02, D01, K01, `network`	
P255 to P314	ts-33-501:2957-3081	Outside the scope	SEPP, roaming intermediaries, attribute protection across trust boundaries and IPUPS: this sub-flow has no roaming leg and no SEPP.
P315 to P318	ts-33-501:3085-3091	No goal stated	The originating-network-id header and its handling by configuration: an operational header requirement, no property of this sub-flow.

Paragraphs	Where	Taken up as	Why not
P319 to P337	ts-33-501:3097-3141	Outside the scope	Monitoring of signalling traffic and security visibility/configurability: operational monitoring, outside the sub-flow and not askable.
P338 to P369	ts-33-501:3149-3215	The prover cannot be asked	Ciphering and integrity algorithm identifiers and algorithm selection: computational strength cannot be asked of a symbolic prover.
P370 to P380	ts-33-501:3219-6548	Outside the scope	5G-RG, NSSAAF and N3 requirements: parts of the system this task leaves out.
P381 to P394	ts-33-501:9656-10917	Outside the scope	EAP-TLS subscriber authentication and IAB requirements: outside the analysed sub-flow.
P395 to P398	ts-33-501:11439-11901	Outside the scope	MBSF, MBSTF and NEF requirements: these functions are not parties of this sub-flow.

The parties and the protected values, one by one

The host listed the 27 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: 22. Without: 5. 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	NWDAF	party	Everything lost by definition	A dishonest NWDAF produces and delivers the data itself; every property asked here is lost by definition and the documents ask none of them under a dishonest producer.
S02	NF service consumer	party	`malicious_nf_resource_id`	
S03	NRF	party	`malicious_nf_resource_id`	
S04	Consumer C	party	`malicious_nf_resource_id`	
S05	Other consumer C'	party	`malicious_nf_resource_id`	
S06	Malicious consumer C'	party	`malicious_nf_resource_id`	
S07	Subscription resource and subscriptionId	value	`malicious_nf_resource_id`	
S08	Notification content	value	`malicious_nf_resource_id`	
S09	notifCorrId and notificURI binding	value	`malicious_nf_resource_id`	
S10	notifCorrId	value	`malicious_nf_resource_id`	
S11	fetchCorrIds	value	`malicious_nf_resource_id`	
S12	subscriptionId	value	`malicious_nf_resource_id`	
S13	Notification data content	value	`malicious_nf_resource_id`	
S14	notifTimestamp	value	Nothing gained	notifTimestamp is only the time the NWDAF prepared the data (s0133); no target rests on its secrecy.
S15	dataNotification	value	`malicious_nf_resource_id`	
S16	dataReports	value	`malicious_nf_resource_id`	

ID	Party or value	Kind	Attacker	Why none
S17	notificURI	value	`malicious_nf_resource_id`	
S18	fetchUri	value	`malicious_nf_resource_id`	
S19	dataSub/anaSub	value	`malicious_nf_resource_id`	
S20	OAuth2 access token	value	Everything lost by definition	A token holder is authorized for the single service scope by definition; token secrecy is asked separately (K01) and no goal is asked under a given token.
S21	SBI session key	value	Everything lost by definition	With the session key of one consumer's TLS session the attacker is that consumer for that session; that session's properties are lost by definition and the documents ask none of them under it.
S22	Subscription resource	value	`malicious_nf_resource_id`	
S23	Collected data and aggregated data	value	`malicious_nf_resource_id`	
S24	OAuth2 token	value	Everything lost by definition	Same as S20: a token holder is authorized for the service scope by definition.
S25	Stored unsent events	value	`malicious_nf_resource_id`	
S26	Notification data	value	`malicious_nf_resource_id`	
S27	Subscription data	value	`malicious_nf_resource_id`	

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_resource_id`	A legal NF with its own credentials: its own access token with the single service scope nnwdaf-datamanagement (s0350), its own TLS session to the NWDAF, and knowledge of another NF's subscriptionId (a plain string in the resource URI, s0226/s0230).	s0350 gives one scope for the whole service and none at resource or operation level; s0230 makes subscriptionId a string identifying the subscription; s0341 lists no application error for a foreign resource. No clause declares subscriptionId or fetchCorrIds confidential.	s0350, s0230, s0226, s0341

ID	`network`	`malicious_nf_resource_id`
A02	Holds	Holds
A03	Holds	Holds
A04	Holds	Holds
D01	Holds	Holds
A08	Holds	Holds
K01	Holds	Holds

ID	network	malicious_nf_resource_id
Z01	Holds	Holds
Z03	Holds	Fails
Z04	Holds	Holds
S05	Holds	Holds
U01	Holds	Holds
U02	Holds	Fails
U03	Holds	Holds

Source and targets

Source: 350 units, 73904 bytes, frozen in source-delivery-01.json. Targets: 13 approved (A02, A03, A04, D01, A08, K01, Z01, Z03, Z04, S05, U01, U02, U03); 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-dc1ce5568bcfec991ad738d2901c

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 48677 tokens in and 5240 out, 256 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
29	A02, A03, A04, D01, A08, K01, Z01, Z03, Z04, S05, U01, U02, U03	plain_primitives	All claims rest only on opaque tokens, session channels and correlation identifiers. The model gives the attacker a symbolic view of TLS (a private channel per connection) and no algebraic laws beyond name equality, so no primitive is simplified in a way that affects the answers.
29	A02, A03, A04, D01, A08, K01, Z01, Z03, U01, U02	many_sessions	The claims concern any number of runs. The malicious NF, its subscription and the fetch/unsubscribe copies are replicated, so ProVerif covers unbounded sessions and clients. The honest run is a single subscription, as the task scopes it.

The analyst's closing note

Established on model.pv (ProVerif, symbolic, any number of sessions): proved -- the notification a consumer accepts is one the NWDAF sent for that subscription with its own notifCorrId (A02, A08, U01); the fetched payload carries the correlators the NWDAF issued (A03, D01); one acceptance per NWDAF send (A04); the consumer's token stays secret (K01); and every served Fetch and DELETE was preceded by an accepted service-scope token (Z01). Every target also has a reachability answer. Attack -- a legal NF that holds its own token and knows a subscriptionId is served another consumer's training data (U02) and deletes that consumer's subscription (Z03). This is the one finding (novelty unclear: s0350 states the single service-wide scope, no clause states the consequence), and a checked fix (creator identity stored and compared) removes both attacks while keeping the normal run. Not settled: Z04, S05 and U03 are answered only by weaker readings, because the prover's tables cannot be emptied here, so nothing after the unsubscribe is decided; and A02/A03/A04/D01/A08/U01 hold only because the model keeps the notification endpoint and the correlators from other NFs, which the specification nowhere requires. Blocking the rest: a stateful model of the subscription record (Tamarin+ Agent) for the after-termination claims, and a model in which a legal NF learns another consumer's notificURI and notifCorrId for the sender-identity claims.