Security analysis of MFAF configured notification and fetch

This report in another language: 中文.

19.7.0 · ProVerif+ Agent report

Documents	5G System; Messaging Framework Adaptor Services; Stage 3 (protocol definition)
Properties examined	13: 3 hold, 10 fail (11 attacks)
Findings	4

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

TS 29.522 has the Messaging Framework Adaptor Function hold data and analytics collected elsewhere and hand them to consumers. The DCCF configures the MFAF with the address of an analytics consumer, the MFAF notifies that consumer with a fetchInstruction, and the consumer fetches the data with the fetch correlation identifiers it was given. The MFAF holds the configurations and the data; the DCCF holds an access token for the configuration service; the analytics consumer holds a token for the fetch service and the notification it received; the NRF issues the tokens.

When the exchange ends the consumer has the data for the identifiers it was notified with, and can check that they belong to the configuration it knows. The MFAF knows that the notification and the fetch are authorized by tokens of the two services and that the identifiers are its own, but it has no rule that ties the fetch to the consumer the identifiers were made for.

The parties:

- **DCCF**: NF service consumer that configures the MFAF. Holds: access token for nmfaf-3dadatamanagement.
- **MFAF**: producer of both services. Holds: stored configurations, collected data and analytics.
- **analytics consumer**: notification endpoint of the configuration. Holds: access token for nmfaf-3cadatamanagement.

Figure 1 shows the exchange as the documents specify it.

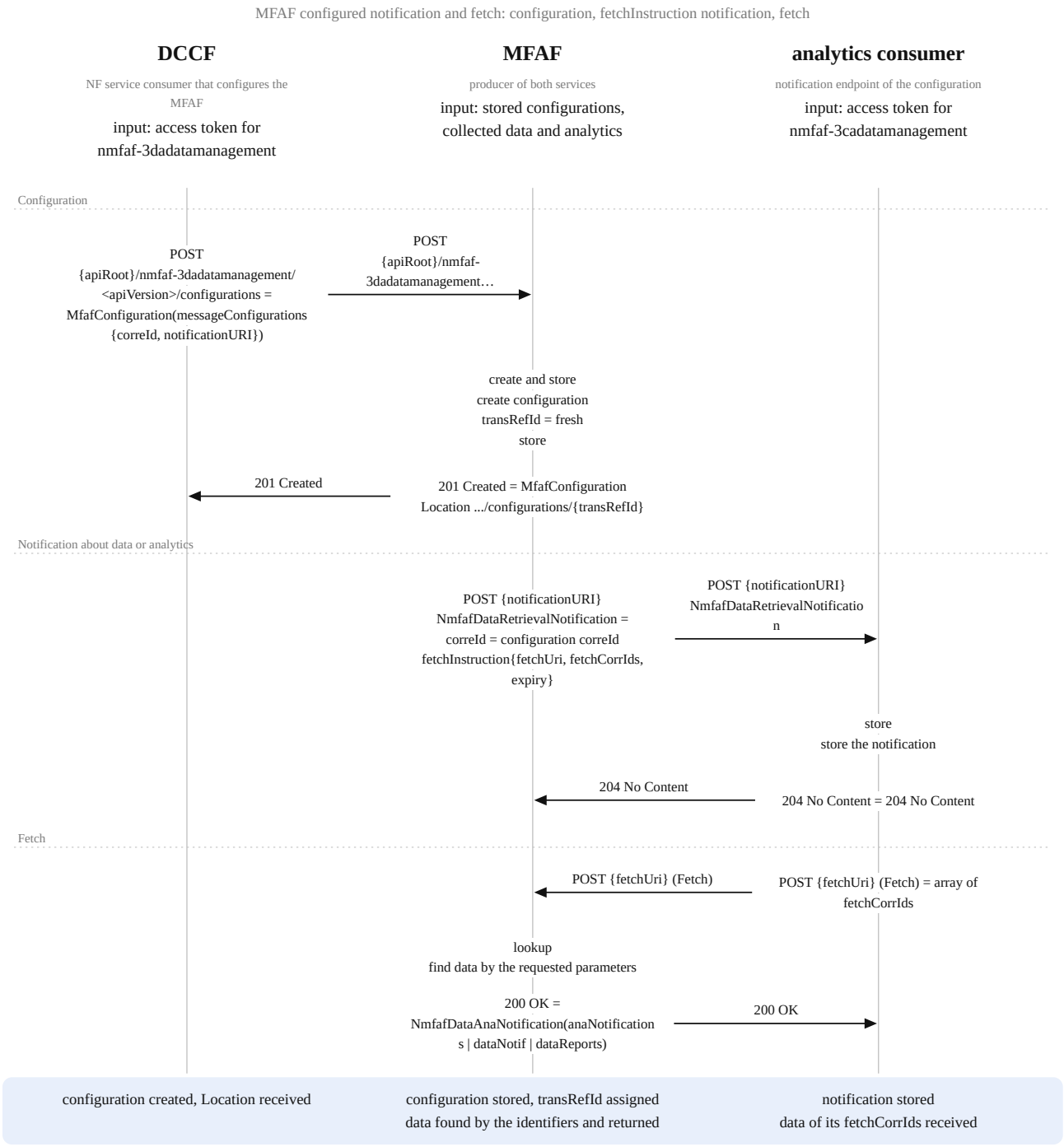


Figure 1. The protocol as specified.

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 4 findings: what follows is not stated in the documents. For 4 of them it is unclear whether the documents state them.

The flow keeps the data from the network attacker and ties the notification to the consumer's own configuration, but it does not tie the fetch to that consumer: the fetchCorrIds are bearer values and the MFAF serves whoever presents them with any token for the service-wide scope. Three weaknesses follow from the specification as written: possession of a notification's identifiers is enough to fetch its data (A03, A08, Z03, K02); the expiry deadline and Deconfigure revoke nothing, so old identifiers keep working (Z04); and an update of the configuration never reaches the fetchInstruction the consumer holds, so pre-update data comes back and is accepted as current (S03, S03.13). Of the thirteen targets, Z01, Z01.12 and R03 are established by proof; the others are partly established, with an attack for the part that fails. The results rest on a model that idealises TLS as one authenticated encrypted channel per pair of network functions and that unrolls the configuration lifetime to three fetches; the finding that any NF holding a transRefId can rewrite a configuration was not confirmed by a query and is therefore not reported as a finding.

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 holding a notification's fetchCorrIds fetches its data

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (main:2640, rfc6749:2879)
Value	medium
Properties	A03, A08, A08.11, Z03, K02, S04 (section 3)
Shown by	ProVerif attack queries `a03_owner_ctl`, `k02_rogue_ctl`; Figures 2, 3, 4, 5, 6 and 7

What happens. A Fetch is granted on the presentation of the fetchCorrIds and a token for the service scope; the MFAF is not required to compare the requester with the consumer of the configuration the identifiers belong to. A second NF that holds another consumer's identifiers is therefore served that consumer's data, and if the identifiers later leak they can be redeemed the same way.

What the attacker needs. The correId, fetchUri and fetchCorrIds of one notification, obtained from an SCP that handles or redirects the notification (all parties must trust the SCP, ts-33-501:2921) or from a previous holder of the values, and any access token for the service-wide scope nmfaf-3cadatamanagement, which every authorized NF can obtain.

What the attacker gains. It receives the analytics data or data reports of that notification, since the MFAF returns the stored data for the identifiers without comparing the requester with the configuration.

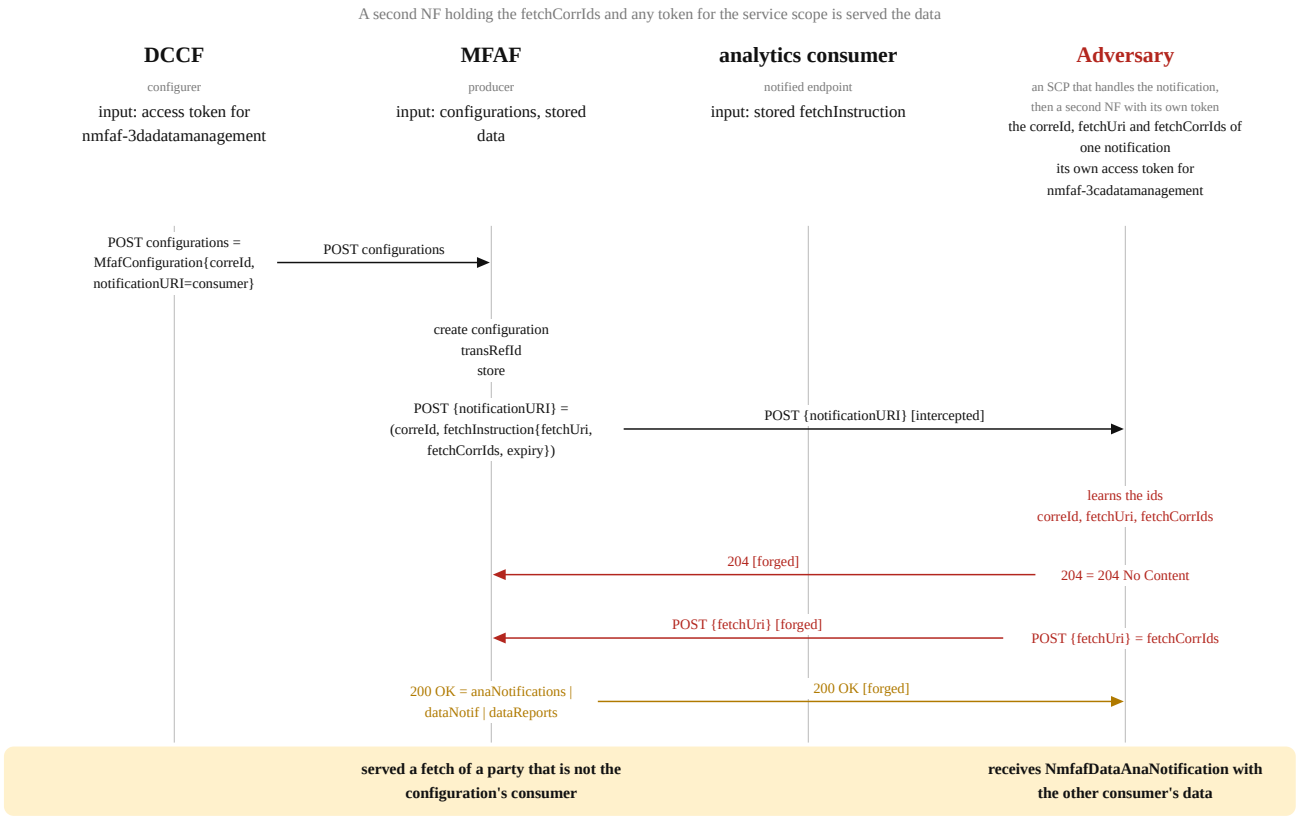


Figure 2. A second NF holding the fetchCorrIds and any token for the service scope is served the data. Property A03; attack query a03_owner_ctl.

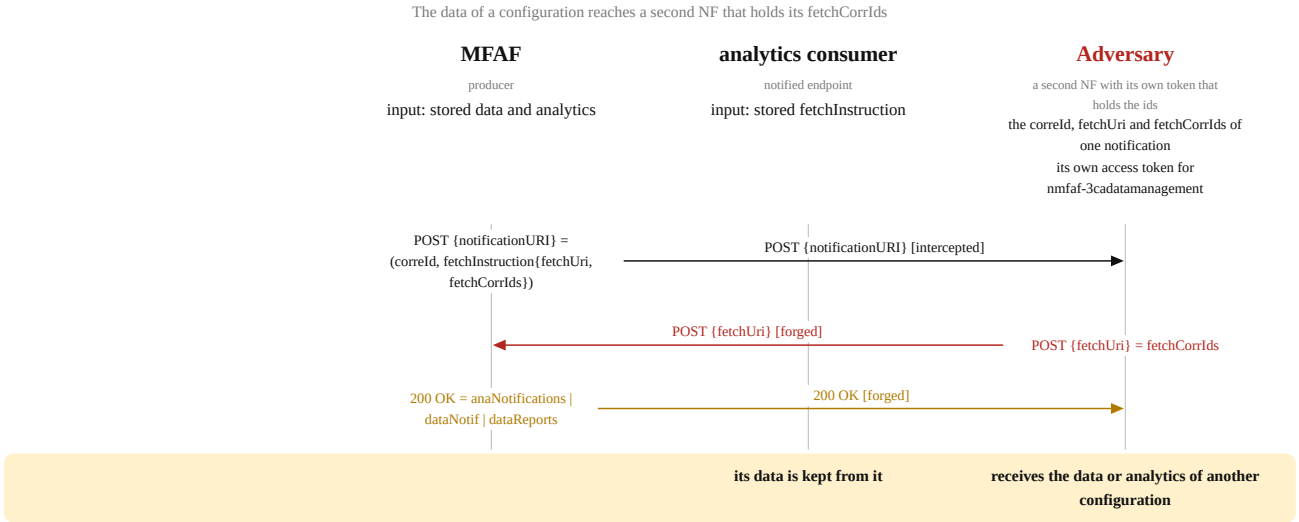


Figure 3. The data of a configuration reaches a second NF that holds its fetchCorrIds. Property K02; attack query k02_rogue_ctl.

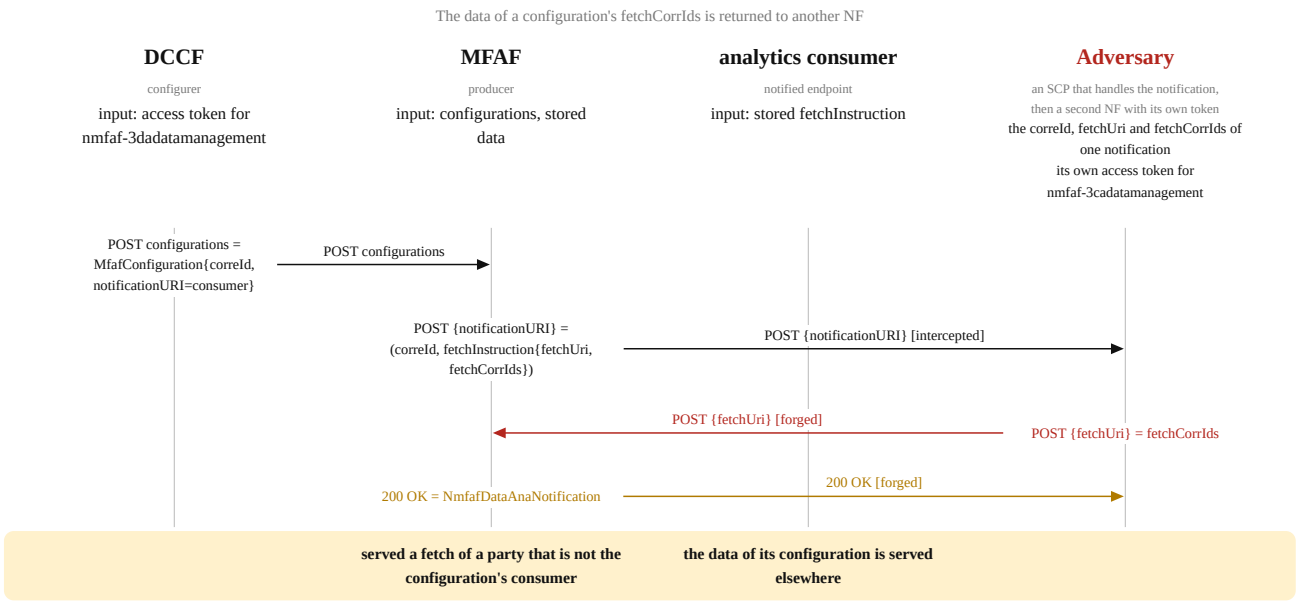


Figure 4. The data of a configuration's fetchCorrIds is returned to another NF. Property A08; attack query a03_owner_ctl.

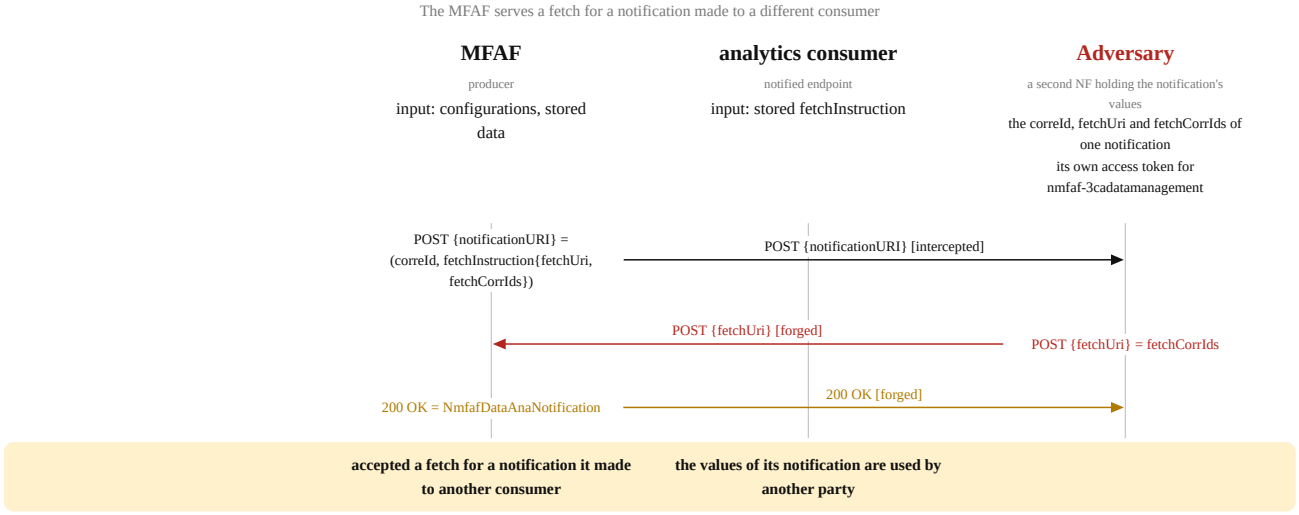


Figure 5. The MFAF serves a fetch for a notification made to a different consumer. Property A08.11; attack query a03_owner_ctl.

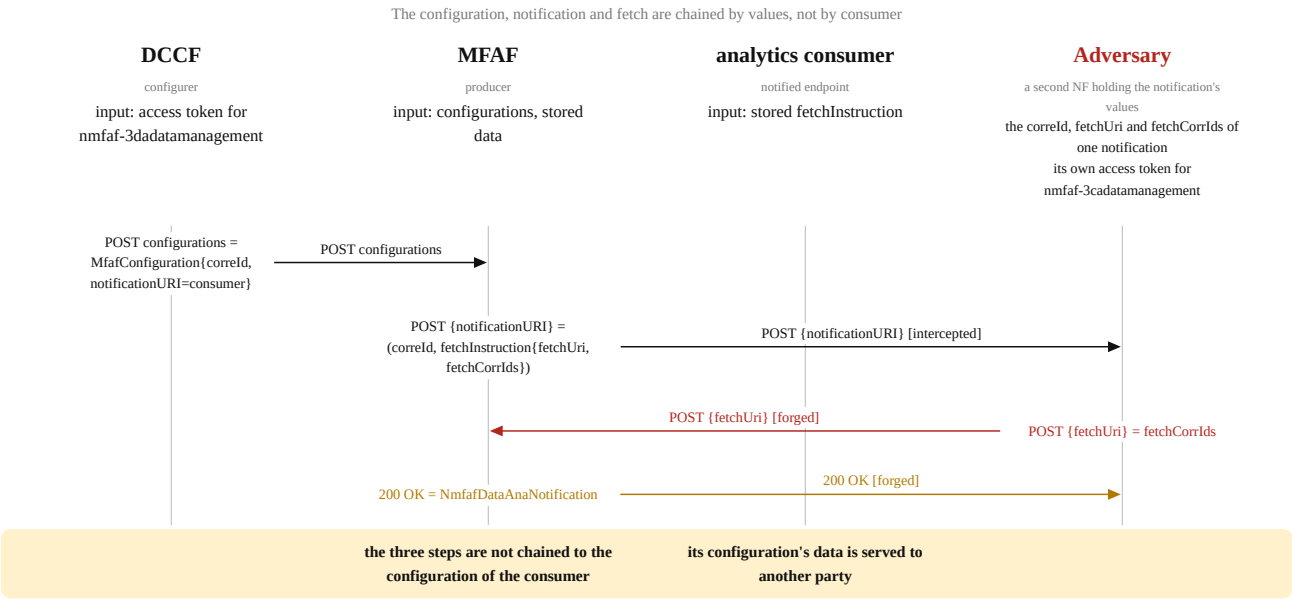


Figure 6. The configuration, notification and fetch are chained by values, not by consumer. Property S04; attack query a03_owner_ctl.

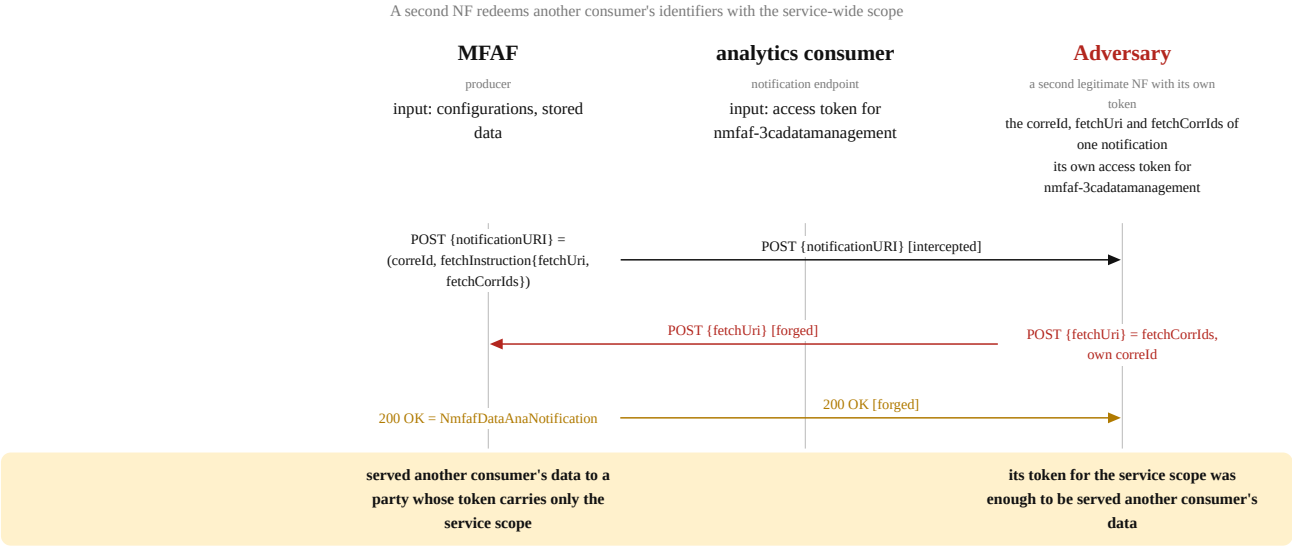


Figure 7. A second NF redeems another consumer's identifiers with the service-wide scope. Property Z03; attack query a03_owner_ctl.

Proposed change. The MFAF shall store, with the fetchCorrIds of a fetchInstruction, the NF service consumer of the configuration that produced them, and shall serve a Fetch only if the subject of the access token is that consumer. As an alternative, make the identifiers single-use and short-lived.

Check of the change with ProVerif. Not checked: No check was run on this recommendation within the session budget, so it is unchecked: the model would first have to store, with each fetchCorrIds, the consumer of the configuration it was made for, and compare it with the token subject on every Fetch.

2.2 The expiry of a fetchInstruction and Deconfigure revoke nothing

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (main:2461)
Value	medium
Properties	Z04 (section 3)
Shown by	ProVerif attack queries `z04_expiry`, `z04_delete`; Figure 8

What happens. The fetchInstruction carries an expiry described as a deadline to fetch the data, and Deconfigure removes the individual configuration, but the MFAF's steps for a Fetch list neither a comparison with the expiry nor a check that the configuration still exists. The same fetchCorrIds therefore keep returning the data after the deadline and after the configuration was deleted.

What the attacker needs. The identifiers of a notification, held by the consumer itself, by a stale instance or by an intermediary; no token beyond the service scope is needed.

What the attacker gains. The data stays retrievable after the deadline the MFAF itself set and after the configuration was deleted, so a stale or disclosed set of identifiers keeps its access for as long as the data is stored.

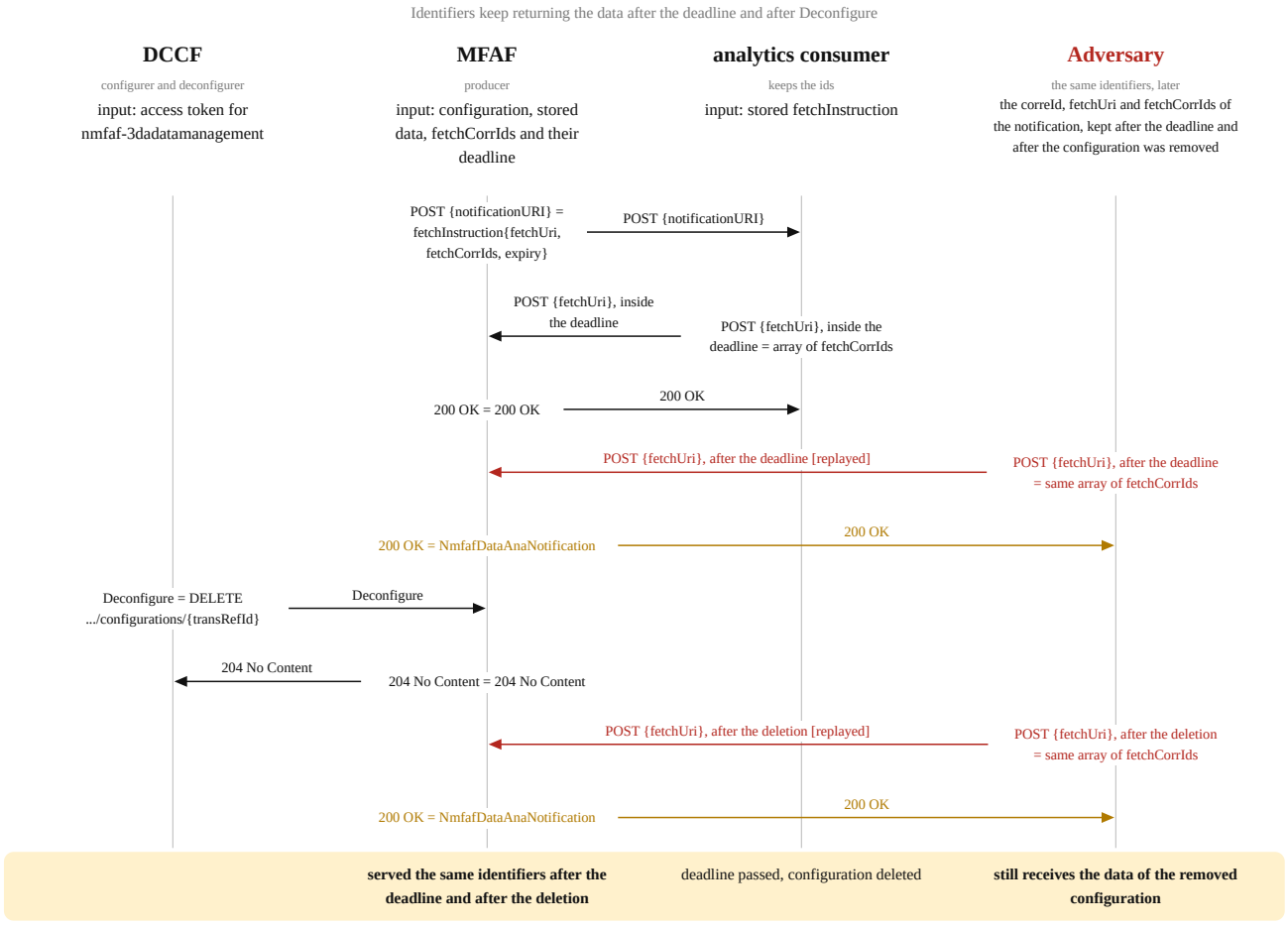


Figure 8. Identifiers keep returning the data after the deadline and after Deconfigure. Property Z04; attack queries z04_delete, z04_expiry.

Proposed change. The MFAF shall, on every Fetch, compare the current time with the expiry of the fetchInstruction to which the fetchCorrIds belong and shall answer with an error once the deadline passed. Deconfigure shall remove the fetchCorrIds and the data of the configuration it removes.

Check of the change with ProVerif. Not checked: No check was run on this recommendation within the session budget, so it is unchecked; a checked version would add the comparison with the expiry and the removal of outstanding identifiers to bounded.pv and rerun the two attack queries.

2.3 An update of the configuration leaves the consumer's fetchInstruction current

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (main:2362)
Value	medium
Properties	S03, S03.13 (section 3)
Shown by	ProVerif attack queries `s03_update`, `s03_accept`; Figures 9 and 10

What happens. An update replaces the stored configuration, but nothing reaches the consumer or the identifiers already issued. The consumer keeps the correId and fetchCorrIds of the old configuration, fetches with them, receives pre-update data carrying the old correId, and accepts it as the data of the configuration it knows, because the source defines no configuration version to compare.

What the attacker needs. Nothing beyond what the consumer already holds: the update is performed by the legitimate configurer, and the consumer's own identifiers are enough.

What the attacker gains. The consumer cannot tell that its instruction belongs to a superseded configuration: it treats pre-update data as the data of the configuration in force, and the MFAF keeps serving it.

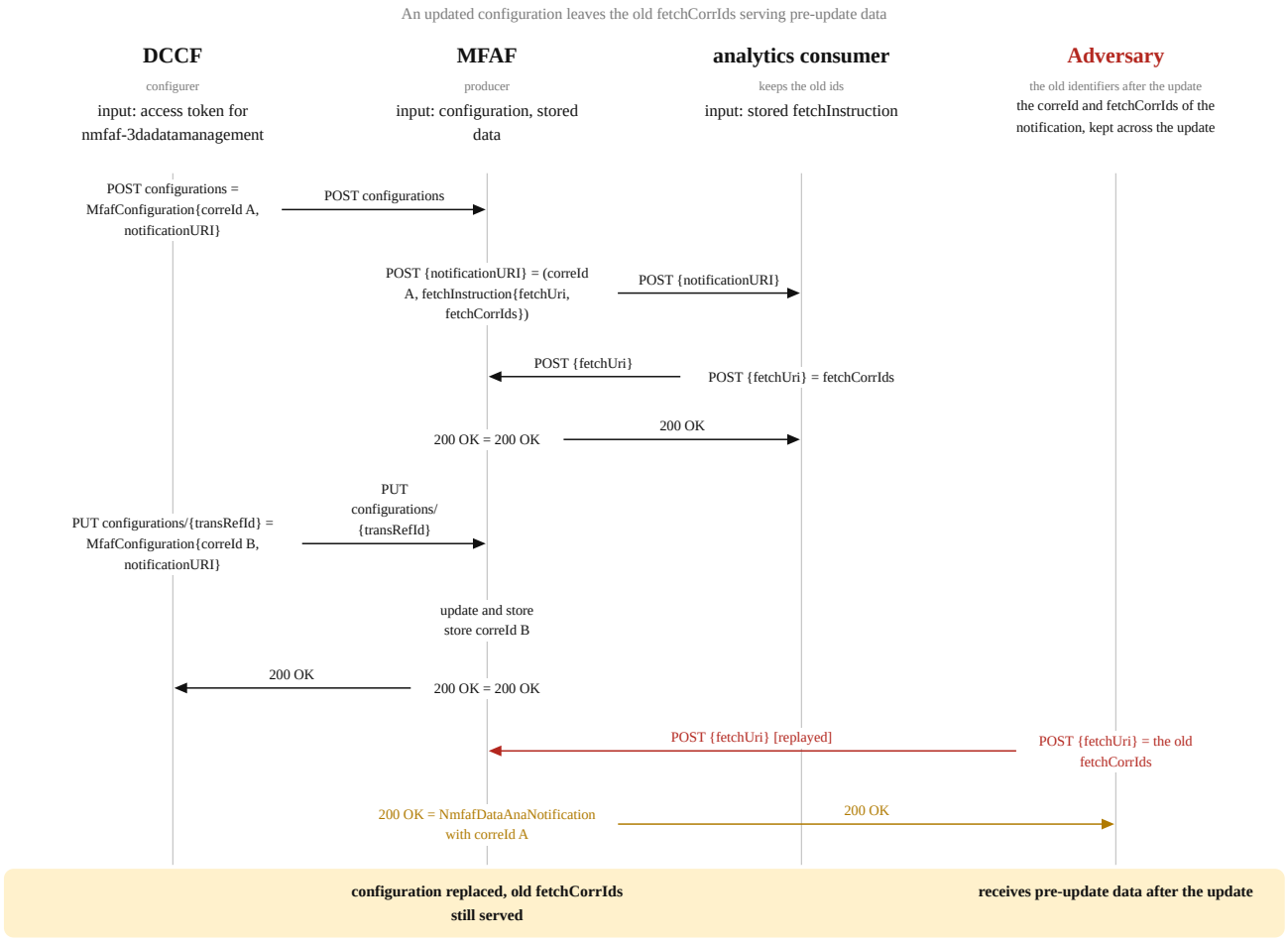


Figure 9. An updated configuration leaves the old fetchCorrIds serving pre-update data. Property S03; attack query s03_update.

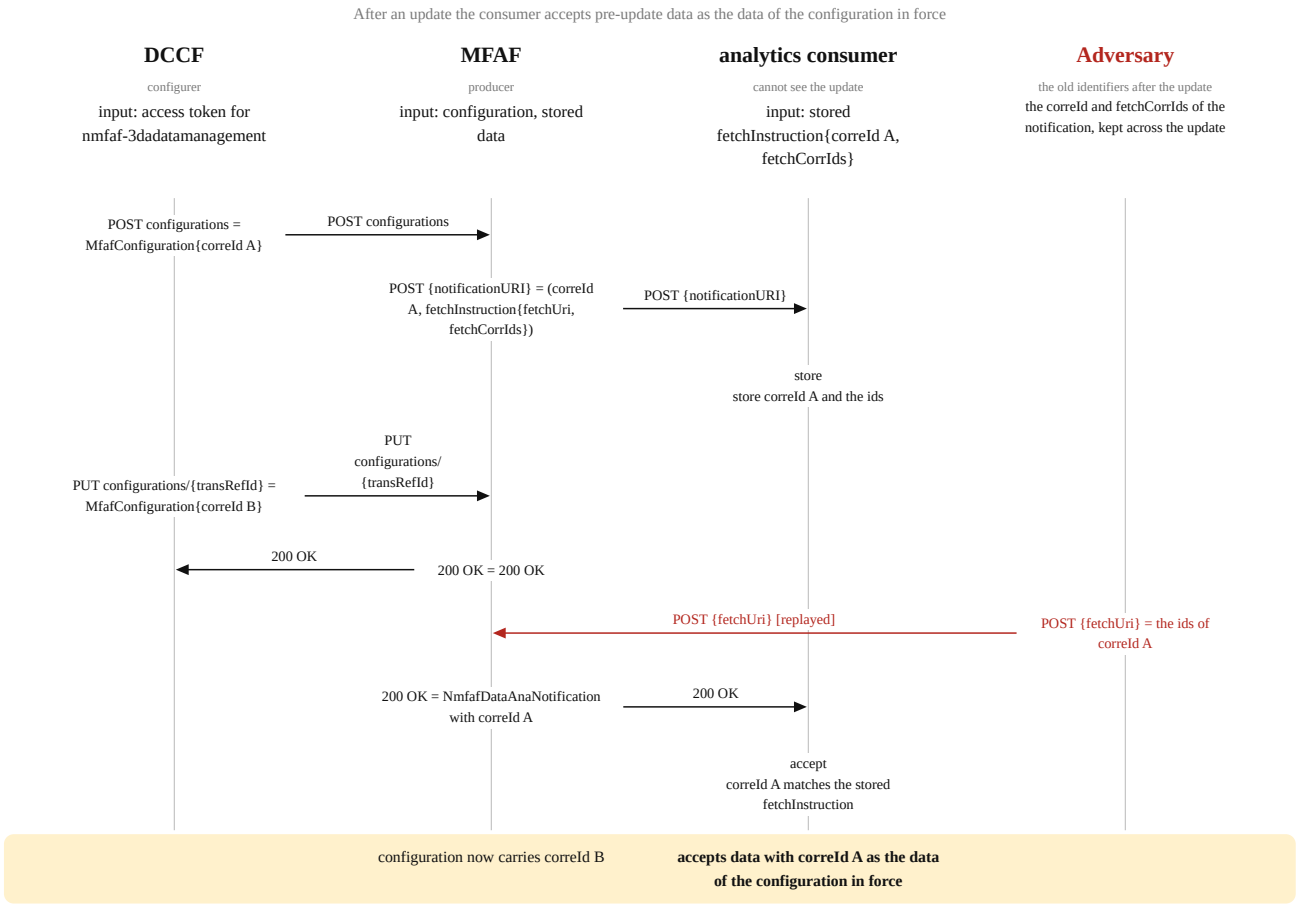


Figure 10. After an update the consumer accepts pre-update data as the data of the configuration in force. Property S03.13; attack query s03_accept.

Proposed change. Carry the configuration's version, incremented by every update, in the fetchInstruction and in the fetch response, and have the consumer accept data only for the version of the configuration it currently holds. Where an update is meant to cancel earlier instructions, state that an update voids the fetchCorrIds of the configuration.

Check of the change with ProVerif. Not checked: No check was run on this recommendation within the session budget; a checked version would add a configuration version to the notification and the response in bounded.pv and rerun the update queries.

2.4 A duplicated notification is accepted and stored again

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (ts-33-501:2865)
Value	low
Properties	R02 (section 3)
Shown by	ProVerif attack query `r02`; Figure 11

What happens. A notification is identified only by its correId and carries no message identity, and the consumer stores every notification it accepts. A second delivery of the same notification is therefore accepted and stored again, and is indistinguishable from a fresh one.

What the attacker needs. A duplicate delivery of one notification, for instance a retry by an SCP that handles the message; the attacker does not need the content of the notification.

What the attacker gains. The consumer stores and acts on the same data or fetchInstruction twice; where the duplicate carries a fetchInstruction, the same data is fetched again.

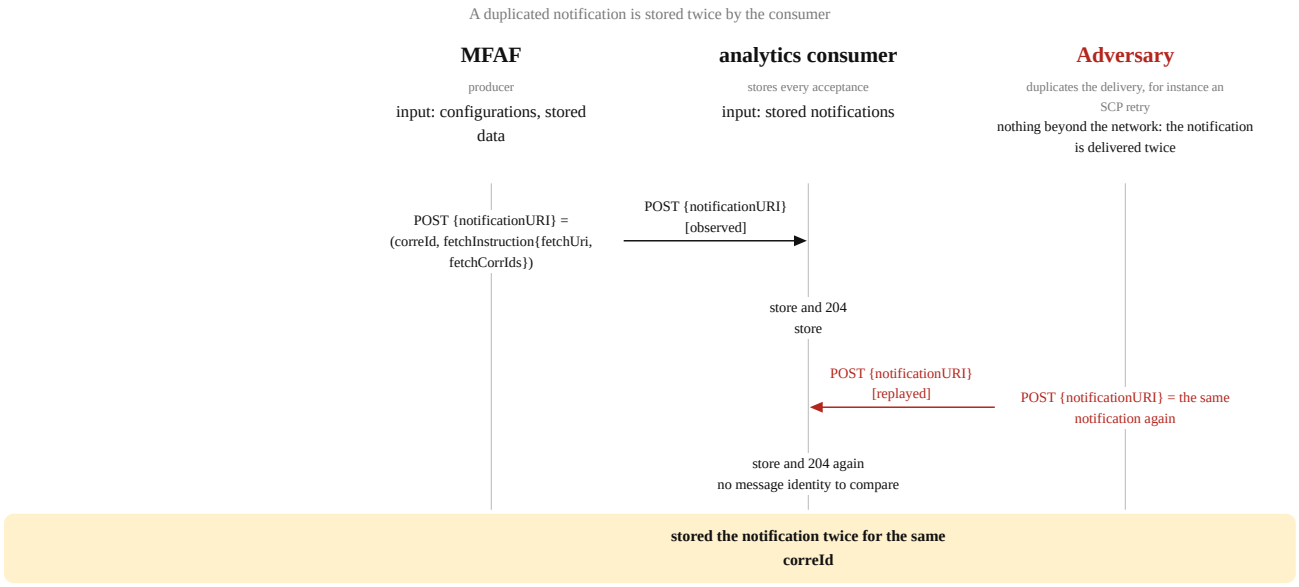


Figure 11. A duplicated notification is stored twice by the consumer. Property R02; attack query r02.

Proposed change. Give the MFAF notification a message identity that the consumer compares with the notifications it stored for that correId, and state that a duplicate is acknowledged with 204 and not stored again.

Check of the change with ProVerif. Not checked: No check was run on this recommendation within the session budget; a checked version would add a message identity to the notification and a comparison with the stored ones in the consumer, in model.pv and over several rounds in bounded.pv.

No figure was given for these attacks of a finding: A03: k02_rogue_ctl, A08.11: k02_rogue_ctl, A08: k02_rogue_ctl, K02: a03_owner_ctl, S03.13: s03_update, S03: s03_accept, S04: k02_rogue_ctl, Z03: k02_rogue_ctl.

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	**Notification and fetch correlation.** Whenever an analytics consumer accepts an MFAF notification, there is a real run of the MFAF whose correId matches the correId in that consumer's configuration; whenever the MFAF accepts a Fetch, there is a real run of that consumer whose fetchCorrIds match the fetchCorrIds the MFAF previously sent in that notification.	no compromise	Fails (1 attack)	medium	unclear
A08	**Fetch uses its own notification's ids.** The fetchUri and fetchCorrIds in a fetchInstruction accepted by a consumer come from the notification that corresponds to its own configuration; the data returned by the MFAF corresponds to the fetchCorrIds carried in the request, not to those of another configuration or another consumer.	no compromise	Fails (1 attack)	medium	unclear

ID	Property	Compromise assumed	Result	Value	In the documents
Z01	**Every operation carries an authorization.** Every time a configuration, notification or Fetch operation is submitted, there must be a permission issued by an authorization entity (the NRF as authorization server) that is valid for that operation; authentication alone is not enough to produce a protected operation.	no compromise	Holds	medium	yes
Z03	**No transfer of tokens or correlation ids.** A consumer's token and correId/fetchCorrIds may be used only for its own configuration and fetch operations; an attacker cannot move one consumer's permission or correlation identifiers to another consumer or another operation.	no compromise	Fails (1 attack)	medium	unclear
Z04	**Expired or revoked ids fetch nothing.** After Deconfigure takes effect, or after the expiry of a fetchInstruction, a new Fetch using the old correId/fetchCorrIds no longer returns data.	no compromise	Fails (2 attacks)	medium	unclear
S03	**No stale data after reconfiguration.** After a configuration update takes effect, a fetchInstruction or old fetchCorrIds based on the old configuration cannot make the MFAF return to the old configuration state or return data under the old configuration.	no compromise	Fails (1 attack)	medium	unclear
S04	**Configuration, notification and fetch chained.** The fetchCorrIds used in the fetch stage come from the notification generated for the same consumer in the notification stage; the correId in the notification stage comes from the configuration established for the same consumer in the configuration stage; there is no cross-consumer or cross-configuration misconnection between the three stages.	no compromise	Fails (1 attack)	medium	unclear
R02	**Notification accepted at most once.** The same logical MFAF notification or the same logical Fetch request is accepted at most once by the receiver within the selected scope.	no compromise	Fails (1 attack)	low	unclear
R03	**Fetch performed at most once.** The same logical fetch operation (one fetch corresponding to one set of fetchCorrIds) commits security-relevant side effects at most once within the selected scope; the request may be retransmitted, and a cached response may be returned.	no compromise	Holds	low	no
K02	**Data visible only to its consumer.** The data or analysis results carried in MFAF notifications and Fetch responses are visible only to the consumer or notification endpoint specified in the configuration; other legitimate NFs and external attackers cannot obtain them.	no compromise	Fails (1 attack)	medium	unclear
A08.11	**Ids bound to the notified consumer.** The correId in a notification accepted by a consumer and the fetchCorrIds in the fetchInstruction correspond to the notificationURI and correId assigned to that consumer in the configuration stage; the fetchCorrIds in a Fetch accepted by the MFAF correspond to the notification it generated for the same consumer.	no compromise	Fails (1 attack)	medium	unclear
Z01.12	**Fetch requires an authorization.** Every time a Fetch operation is submitted, there must be a permission issued by an authorization entity that is valid for that operation; authentication alone is not enough to fetch data.	no compromise	Holds	medium	yes
S03.13	**Old instruction cannot outlive update.** After a configuration update takes effect, a fetchInstruction or old fetchCorrIds based on the old configuration cannot make the MFAF return data under the old configuration, nor make the consumer accept the old data as the data of the new configuration.	no compromise	Fails (1 attack)	medium	unclear

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: binding of notificationURI/correId/fetchCorrIds to the data receiver	Within one consumer the binding holds: the notification it accepts carries the correId of a configuration that names it, and the fetchCorrIds it fetches with come from that notification. Across consumers it does not: the MFAF serves any NF that presents the identifiers, so the binding is to the values, not to the data receiver.	A03, A08, A08.11, K02
Q2: whether another legitimate NF's fetch permission comes from an explicit specification rule	No: the only rule is the service-wide scope nmfaf-3cadatamanagement, which every authorized NF obtains, and the MFAF is not required to compare the requester with the consumer of the fetchCorrIds. A Fetch succeeds against a valid token of that scope plus possession of the identifiers.	Z01.12, Z03, A08
Q3: whether old fetchInstructions and the data of an updated configuration cannot be confused	No: an update replaces the stored configuration but leaves the old fetchCorrIds working, and the consumer has no way to see that its instruction belongs to a superseded configuration, so old data comes back and is accepted as current.	S03, S03.13

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 one authenticated encrypted channel per pair of network functions; the SBI's own replay protection, certificate handling and the SCP and SEPP paths are the baseline of TS 29.500 and TS 33.501 and are outside the model.
- OAuth2 tokens are modelled as a signature of the authorization server checked with its provisioned public key; token lifetime, audience and the token request are outside the model.
- The configuration lifetime is unrolled to create, notify, fetch, update, fetch, deconfigure, fetch in bounded.pv; what happens beyond three fetch rounds is not covered, and model.pv leaves the update and the deactivation out entirely.
- The MFAF's internal association between a configuration, a notification and the stored data is not specified by the source; the model gives one notification per configuration and looks the data up by the identifiers.
- The recovery of the identifiers by a third party (an SCP, a redirect, a stale instance) is modelled as a role that discloses them; whether such disclosure is possible in a deployment is not settled by the source.
- The update and delete of an Individual MFAF Configuration are identified only by the transRefId and the source lists no ownership check; this candidate from the state review is not confirmed by a query and is not reported as a finding.
- No availability, timing, cost or liveness property is analysed, and the cryptographic strength of the algorithms is not considered.

What the host records:

- The answers of Z04, S03, S03.13 rest on an operation the model simplifies (Appendix D).
- Proved without a control query (Appendix C): A03: a03_notify, A08: a08_cfg, Z01: z01_cfg, Z03: a03_owner, S04: a08_cfg, A08.11: a03_notify, A08.11: a08_cfg. 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: Notification and fetch correlation

Whenever an analytics consumer accepts an MFAF notification, there is a real run of the MFAF whose correId matches the correId in that consumer's configuration; whenever the MFAF accepts a Fetch, there is a real run of that consumer whose fetchCorrIds match the fetchCorrIds the MFAF previously sent in that notification.

The notification and the fetch are bound by their values: a notification the consumer accepts always comes from a fetchInstruction the MFAF really sent for a configuration naming that consumer, and a fetch the MFAF accepts always refers to such a notification. What is not bound is the identity of the party: whoever presents the fetchCorrIds is served, and the model produces a trace in which a second NF is served for another consumer's notification.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: the correspondence between the three operations holds against the network attacker, but the party that owns the fetchCorrIds is not identified, which is what makes the target relevant.
- Stated in the documents: unclear. Source text it rests on: ts-33-501:2873, ts-33-501:2921, s0120, s0121, s0122, s0124, s0140, s0395, s0398
- Rests on: TLS is modelled as one authenticated encrypted channel per pair of NFs, so the network attacker can neither read nor originate a message addressed to an NF; the notification is identified only by values the MFAF chooses; no message identity is modelled; the second NF can hold the ids only through the modelled disclosure role
- Open: whether a notification's correId, fetchUri and fetchCorrIds can reach a party other than the notified consumer is not settled by the source
- Next step: a statement in TS 29.522 or TS 33.501 on who may present fetchCorrIds, and a Tamarin+ Agent model of the SCP path, would settle it

A08: Fetch uses its own notification's ids

The fetchUri and fetchCorrIds in a fetchInstruction a consumer accepts come from the notification for its own configuration; the data the MFAF returns correspond to the fetchCorrIds carried in the request, not to those of another configuration or another consumer.

The consumer can rely on its fetchUri and fetchCorrIds coming from its own configuration's notification, and on the response carrying the data of the ids it sent. But the MFAF returns the data of a set of ids to any NF that holds them, so the instruction is not bound to the consumer it was made for.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: the consumer side holds; the MFAF side fails as soon as a party other than the consumer holds the ids.
- Stated in the documents: unclear. Source text it rests on: rfc6749:2879, ts-33-501:2921, s0366, s0374, s0376, s0398, s0429
- Rests on: TLS is modelled per pair of NFs; the MFAF looks the data up by the identifiers only, as the source describes; the disclosure of the ids to a second NF is modelled as a role, not derived from the source

- Open: whether the source intends fetchCorrIds to be one-time or bound to a consumer is not said
- Next step: an ownership rule in the source, or a model in which the MFAF stores the consumer of each fetchCorrIds, would settle it

A08.11: Ids bound to the notified consumer

The correId in a notification a consumer accepts and the fetchCorrIds in the fetchInstruction correspond to the notificationURI and correId assigned to that consumer during configuration; the fetchCorrIds in a Fetch the MFAF accepts correspond to the notification it generated for the same consumer.

The notifications a consumer accepts carry the correId of the configuration that names it as notificationURI, so configurations and consumers are not crossed on the notification path. The MFAF, however, accepts a Fetch for its notification from any NF holding the identifiers.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: the second half of the claim, that the MFAF serves only the consumer it notified, is falsified.
- Stated in the documents: unclear. Source text it rests on: ts-33-501:2921, s0140, s0395, s0268, s0398
- Rests on: TLS per pair of NFs; no storage of notifications in the consumer beyond storing the values; the disclosure of the ids by an endpoint or intermediary
- Open: how a third party obtains correId and fetchCorrIds is outside the source units selected
- Next step: checking the SCP and redirect paths of TS 29.500 clause 6.10.9 against this flow would settle it

Z01: Every operation carries an authorization

Every configuration, notification or Fetch operation that is submitted must have a valid authorization for that operation issued by the authorization subject (the NRF as authorization server); identity authentication alone is not enough to produce a protected operation.

Every Configure the MFAF stores and every Fetch it serves is preceded by an access token the authorization server issued to that NF for that service's scope; an NF that holds a token for another service is never served. Identity over TLS alone is not enough.

- Result: Holds; answers what is asked: yes; usable
- Value medium: confirms the mechanism the source describes, on a model that idealises the token as a signature; the interesting limit is that the scope is service-wide, which is Z03's business.
- Stated in the documents: yes. Where the documents say it: main:1982, main:2634, s0304, s0305, s0426, s0427
- Rests on: the token is modelled as a signature of the authorization server, with its public key provisioned in the MFAF; the subject of the token is compared with the identity the request carries; token lifetime and audience are not modelled
- Next step: nothing; the token's own freshness and audience are the OAuth2 baseline, which the task leaves out

Z01.12: Fetch requires an authorization

Every Fetch operation that is submitted must have a valid authorization for that operation issued by the authorization subject; identity authentication alone is not enough to fetch data.

A Fetch is performed only for a party whose token carries nmfaf-3cadatamanagement; an NF authenticated by TLS with a token for another service is refused, and a legitimate NF with the service-wide scope is served. That scope is the whole of the authorization the Fetch has.

- Result: Holds; answers what is asked: yes; usable
- Value medium: it shows that the only condition on a Fetch is a token for the service, which is the mechanism the single scope provides.
- Stated in the documents: yes. Where the documents say it: main:2634, main:2640, s0426, s0429
- Rests on: the token is a signature verified with the authorization server's key; no token expiry, audience or re-use rule is modelled; the pairwise TLS channel gives the sender's identity
- Next step: nothing for the authorization question; whether the scope should be finer is a design question the source answers in the negative

Z03: No transfer of tokens or correlation ids

A consumer's token and correId/fetchCorrIds can only be used for its own configuration and fetch operations; an attacker cannot move one consumer's authorization or correlation identifiers to another consumer or another operation.

Neither the token nor the correId can be moved to another consumer: without the fetchCorrIds nothing is served, and a second legitimate NF with the service-wide scope is not served for ids it does not hold. Once it holds the ids of another consumer's notification it is served that consumer's data, because the MFAF never compares the requester with the configuration.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: transferring authority to another NF by moving the fetchCorrIds is a weakness of the flow as specified; the documents supply the ingredients without stating it.
- Stated in the documents: unclear. Source text it rests on: main:2640, rfc6749:2879, s0040, s0395, s0398, s0429
- Rests on: the ids reach the attacker only through the modelled disclosure; TLS per pair of NFs; no per-resource scope in the source, so no finer authorization is modelled
- Open: the source does not say whether the MFAF must compare the requester with the consumer of the fetchCorrIds, nor whether these values are secret or single-use
- Next step: a rule binding fetchCorrIds to the consumer, or accepting them as bearer values, would settle it; Tamarin+ Agent can model the SCP path that discloses them

Z04: Expired or revoked ids fetch nothing

After Deconfigure takes effect, or after the expiry of fetchInstruction lapses, a new Fetch using the old correId/fetchCorrIds no longer returns data.

The deadline carried in the fetchInstruction is not a deadline in practice: the MFAF finds the data by the requested identifiers and answers, so the same identifiers keep returning the data after the deadline and after the configuration was removed by Deconfigure. Nothing in the source invalidates the fetchCorrIds already handed out.

- Result: Fails (2 attacks); answers what is asked: yes; usable
- Value medium: it shows that a field the source defines as a deadline has no step that enforces it, with the exact conditions under which the data stays fetchable.
- Stated in the documents: unclear. Source text it rests on: main:2461, s0398, s0076, s0122, s0123, s0124
- Rests on: the configuration lifetime is unrolled to three fetches (bounded.pv); beyond them nothing is claimed; authorization is left out of bounded.pv; the deadline is modelled as one step, so the second and third fetches fall after it

- Open: the source never says what the MFAF must do when a fetch arrives after the expiry or after the configuration was deleted
- Next step: a sentence in clause 4.3.2.2 on the expiry check and on the effect of Deconfigure on outstanding fetchCorrIds would settle it; a bounded model of more fetch rounds would then show the fix

S03: No stale data after reconfiguration

After a configuration update takes effect, a fetchInstruction or old fetchCorrIds based on the old configuration must not make the MFAF return to the old configuration state or return data under the old configuration.

An update replaces the stored configuration but not the identifiers already handed out: the old fetchInstruction keeps working and returns the pre-update data with the pre-update correId. Only a new notification, with new fetchCorrIds, would move the consumer on.

- Result: Fails (1 attack); answers what is asked: yes; usable
- Value medium: the update path is where a stale value can survive, and the source gives no rule for it.
- Stated in the documents: unclear. Source text it rests on: main:2362, s0059, s0060, s0062, s0124, s0395
- Rests on: bounded.pv unrolls one configuration through create, notify, fetch, update, fetch, deconfigure, fetch; no configuration version exists in the source, so none is modelled; authorization left out of bounded.pv
- Open: whether an update is meant to cancel outstanding fetchInstructions is not stated
- Next step: either a configuration version carried into the notification and the Fetch, or an explicit rule that an update voids earlier fetchCorrIds, would settle it

S03.13: Old instruction cannot outlive update

After a configuration update takes effect, a fetchInstruction or old fetchCorrIds based on the old configuration must not make the MFAF return data under the old configuration, nor make the consumer accept the old data as data of the new configuration.

After an update the consumer keeps the correId and fetchCorrIds of the old configuration, fetches with them, receives the old data with the old correId, and accepts it as the data of the configuration it knows. The response carries the correId of the configuration the identifiers belong to, which is exactly the value the consumer already has.

- Result: Fails (1 attack); answers what is asked: yes; usable
- Value medium: it shows that pre-update data is indistinguishable from current data for the consumer, which the source does not address.
- Stated in the documents: unclear. Source text it rests on: main:2362, s0395, s0124, s0059
- Rests on: one update and one further fetch are unrolled; the consumer compares the response with the correId of its stored fetchInstruction, as in s0395; authorization left out of bounded.pv
- Open: the consumer is given no way to learn that the configuration behind its fetchInstruction changed
- Next step: a configuration version, or a new notification after every update, would let the consumer tell the difference

S04: Configuration, notification and fetch chained

The fetchCorrIds used in the fetch phase come from the notification generated for the same consumer in the notification phase; the correId of the notification phase comes from the configuration established for the same consumer in the configuration phase; no cross-consumer or cross-configuration misattachment exists between the three phases.

Within one consumer the three steps are chained: the notification carries the correId of the configuration that names it, and the data follows the identifiers the consumer sent. Across consumers the chain is only the values: a party holding another consumer's identifiers is served, so the chain is not tied to the configuration.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: the chain holds inside a configuration and breaks between configurations, which is the gap Z03 and A08 name.
- Stated in the documents: unclear. Source text it rests on: main:2362, s0035, s0047, s0052, s0122, s0124, s0395
- Rests on: TLS per pair of NFs; the MFAF is modelled per configuration, with one notification per configuration; the disclosure of the ids
- Open: the internal association between a configuration, its notification and its data is not specified
- Next step: a sentence on how the MFAF associates fetchCorrIds with a configuration would settle it; the model could then store that association

R02: Notification accepted at most once

The same logical MFAF notification or the same logical Fetch request is accepted by the receiver at most once within the selected scope.

A notification has no identity other than its correId, so a second delivery of the same notification is accepted and stored again. The source does not settle whether such a duplicate can occur: SBI signalling shall support replay protection, which would stop a network replay but leaves a retry by a party that handles the message.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value low: the flow defines no duplicate detection of its own, but whether the failure can be triggered depends on the transport layer, which the task leaves out.
- Stated in the documents: unclear. Source text it rests on: ts-33-501:2865, s0140, s0395, s0148, s0149
- Rests on: the model's TLS abstraction does not provide the SBI replay protection the source requires, so the trace reaches the second acceptance by a replay on the model's channel; the consumer stores the notification and compares nothing
- Open: whether a duplicate delivery can happen in a deployment (an SCP retry) is outside the source units selected
- Next step: a model of the SCP or of the HTTP layer would say whether a duplicate can be caused; the API itself gives no duplicate detection

R03: Fetch performed at most once

The same logical fetch operation (one fetch corresponding to the same set of fetchCorrIds) commits at most one security-relevant side effect within the selected scope; the request may be retransmitted and may return a cached response.

A fetch reads stored data and changes nothing, so repeating it returns the same data and no state of the MFAF moves. The source gives the operation no side effect, which is what the claim asks.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: the property holds because the operation is a read; it is worth recording only as the counterpart of the expiry gap.
- Stated in the documents: no. Source text it rests on: s0122, s0124, s0374, s0376

- Rests on: one fetch per configuration is modelled in model.pv, so the at-most-once reading holds by construction there; repeated fetches are the subject of Z04 and S03 in bounded.pv
- Open: how many times a fetch may be repeated is not stated
- Next step: nothing for the property as written; a repeated fetch after the deadline is Z04's business

K02: Data visible only to its consumer

The data or analytics carried in the MFAF notification and the Fetch response are visible only to the consumer specified in the configuration or to the notification endpoint; other legitimate NFs and external attackers cannot obtain them.

The network never obtains the data or analytics: the notification and the fetch response are protected end to end between the MFAF and the consumer. A second legitimate NF with the service scope obtains them too as soon as it holds the identifiers of that notification; without them the scope gives it nothing.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: the confidentiality of the data depends entirely on the confidentiality of the fetchCorrIds, which the source never requires.
- Stated in the documents: unclear. Source text it rests on: ts-33-501:2915, s0124, s0140, s0376, s0429
- Rests on: TLS per pair of NFs and the absence of other channels; the data is opaque, so no inference from its contents is modelled; the disclosure of the ids to the second NF
- Open: the source says nothing about who may redeem fetchCorrIds, so the confidentiality of the data rests on their confidentiality
- Next step: binding the identifiers to a consumer, or making them single-use, would make the answer yes

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_notify	1 of 2	a03_owner_ctl (depends_on_ambiguity)	2 proved, 1 attack
A08	proved		a08_cfg	1 of 2	a03_owner_ctl (depends_on_ambiguity)	2 proved, 1 attack
Z01	proved		z01_fetch	2 of 3		3 proved
Z03	proved		a03_owner	1 of 2	a03_owner_ctl (depends_on_ambiguity)	2 proved, 1 attack
Z04	none				z04_expiry (holds_in_source); z04_delete (holds_in_source)	2 attack
S03	none				s03_update (holds_in_source)	1 attack
S04	proved		a08_cfg	1 of 2	a03_owner_ctl (depends_on_ambiguity)	2 proved, 1 attack
R02	none				r02 (depends_on_ambiguity)	1 attack
R03	proved		r03	checked		1 proved
K02	proved		k02_secret	checked	k02_rogue_ctl (depends_on_ambiguity)	2 proved, 1 attack
A08.11	proved		a03_notify		a03_owner_ctl (depends_on_ambiguity)	2 proved, 1 attack
Z01.12	proved		z01_fetch	checked		2 proved
S03.13	none				s03_accept (holds_in_source)	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	When the analytics consumer accepts an MFAF notification with a correId and fetchCorrIds, the MFAF really sent that notification with those values to it.	`network`	proved	a03_notify	none
A03	The party served by a Fetch is the consumer of the configuration; the attacker additionally holds one notification's correId, fetchUri and fetchCorrIds.	`ids_disclosed`	attack (depends_on_ambiguity)	a03_owner_ctl (the control of a03_owner)	
A03	When the MFAF accepts a Fetch with fetchCorrIds, it earlier sent a notification carrying those fetchCorrIds for the configuration whose consumer it notifies.	`network`	proved	a03_fetch	checked
A08	The notification a consumer accepts carries the correId of a configuration that names that consumer as notificationURI.	`network`	proved	a08_cfg	none
A08	The MFAF returns the data only to the consumer named in the configuration of those fetchCorrIds; the attacker additionally holds one notification's correId, fetchUri and fetchCorrIds.	`ids_disclosed`	attack (depends_on_ambiguity)	a03_owner_ctl (the control of a03_owner)	

target	claim	attacker	answer	query	control
A08	The data a party receives corresponds to the fetchCorrIds that party put in its Fetch, not to another party's.	`network`	proved	a08_data	checked
Z01	Every Fetch the MFAF accepts is preceded by an access token the authorization server issued to the requester for the fetch service scope.	`network`	proved	z01_fetch	checked
Z01	Every Configure the MFAF stores is preceded by an access token issued to the configuring NF for the configuration service scope.	`network`	proved	z01_cfg	none
Z01	A legitimate NF with its own token cannot configure or fetch without that token being checked for the operation's scope and its own subject.	`rogue_nf`	proved	z01_fetch	checked
Z03	A second legitimate NF with its own credentials and the service-wide scope cannot turn them into another consumer's data: it is served only ids of a notification it was given.	`rogue_nf`	proved	a03_owner	none
Z03	A consumer's correId and fetchCorrIds cannot be redeemed by another NF for the same data; the attacker additionally holds one notification's correId, fetchUri and fetchCorrIds.	`ids_disclosed`	attack (depends_on_ambiguity)	a03_owner_ctl (the control of a03_owner)	
Z03	A token, correId or fetchCorrIds cannot be used for another consumer's configuration or another operation: without the values a party is not served.	`network`	proved	a03_owner	checked
Z04	After the deadline carried in the fetchInstruction, a Fetch with the old fetchCorrIds returns no data.	`network`	attack (holds_in_source)	z04_expiry	
Z04	After the configuration is deleted, a Fetch with the old correId and fetchCorrIds returns no data; the attacker additionally holds those old identifiers.	`ids_disclosed`	attack (holds_in_source)	z04_delete	
S03	An old fetchInstruction's ids cannot be used after the configuration was updated to return that configuration's earlier data; the attacker additionally holds the old identifiers.	`ids_disclosed`	attack (holds_in_source)	s03_update	
S04	The notification carries the correId of the configuration that names this consumer, so the configuration and notification steps are not crossed between consumers.	`network`	proved	a08_cfg	none
S04	A consumer different from the one the configuration names is never served the data of that configuration; the attacker additionally holds one notification's correId, fetchUri and fetchCorrIds.	`ids_disclosed`	attack (depends_on_ambiguity)	a03_owner_ctl (the control of a03_owner)	
S04	The data received follows the fetchCorrIds the same party sent, so the fetch phase is not crossed with another consumer's fetch.	`network`	proved	a08_data	checked

target	claim	attacker	answer	query	control
R02	One logical MFAF notification is accepted at most once: a second delivery is a second acceptance.	`network`	attack (depends_on_ambiguity)	r02 (the control of r03)	
R03	One logical fetch has at most one security-relevant effect: it reads stored data and changes nothing, so a retransmission returns the same data.	`network`	proved	r03	checked
K02	The external attacker never learns the analytics data or analysis results carried in the notification and the fetch response.	`network`	proved	k02_secret	checked
K02	No other legitimate NF obtains the data of a fetchCorrIds; the attacker additionally holds one notification's correId, fetchUri and fetchCorrIds.	`ids_disclosed`	attack (depends_on_ambiguity)	k02_rogue_ctl (the control of a03_owner)	
K02	Another legitimate NF obtains the data only if that notification's fetchCorrIds were disclosed to it: the service scope alone does not give it the data.	`rogue_nf`	proved	a03_owner	checked
A08.11	The correId and fetchCorrIds a consumer accepts come from the MFAF's notification for the configuration whose notificationURI is that consumer.	`network`	proved	a03_notify	none
A08.11	The notification accepted corresponds to a configuration naming that consumer and that correId.	`network`	proved	a08_cfg	none
A08.11	The MFAF accepts a Fetch only for the notification it made to that same consumer; the attacker additionally holds one notification's correId, fetchUri and fetchCorrIds.	`ids_disclosed`	attack (depends_on_ambiguity)	a03_owner_ctl (the control of a03_owner)	
Z01.12	An NF that authenticates over TLS but holds only a token for another service is never served a Fetch: identity authentication alone does not retrieve data.	`network`	proved	z01_noauth	checked
Z01.12	A Fetch is performed only for a party whose token carries the fetch service scope: the service-wide scope the NRF issues is necessary and sufficient.	`rogue_nf`	proved	z01_fetch	checked
S03.13	Old data cannot be accepted as the data of the updated configuration; the attacker additionally holds the old correId and fetchCorrIds.	`ids_disclosed`	attack (holds_in_source)	s03_accept	

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-20261114T230127, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 43, prover 2 s, wall 16 min, cost 0.130 USD.

ProVerif: 59 proofs run, 58 decided, 1 without an answer, 2 s, 38 answers recorded.

What was analysed

Protocol	MFAF configured notification and fetch
Version	19.7.0
Analysed	One DCCF configurator, MFAF, one analytics consumer; aggregated data has been collected, configuration → fetchInstruction notification → Fetch with fetchCorrIds → data response, a bounded transfer. The configuration procedure must be included; the nonexistent 3ca Subscribe operation must not be invented; upstream collection and complex processing are excluded.
Not analysed	The symbolic model does not prove the computational security or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive-case 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. Monotonic tables cannot model real deletion; the finite-state lifecycle must be clearly defined. The goal is the specified API sub-flow, not complete 5G security.
Source text	477 numbered passages (195817 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	Nmfaf_3daDataManagement Service
main	4.2.1	Service Description
main	4.2.1.1	Overview
main	4.2.1.2	Service Architecture
main	4.2.1.3	Network Functions
main	4.2.1.3.1	Messaging Framework Adaptor Function (MFAF)
main	4.2.1.3.2	NF Service Consumers
main	4.2.2	Service Operations
main	4.2.2.1	Introduction
main	4.2.2.2	Nmfaf_3daDataManagement_Configure service operation
main	4.2.2.2.1	General
main	4.2.2.2.2	Initial configuration of the MFAF
main	4.2.2.2.3	Update the configuration of the MFAF

Document	Section	Title
main	4.2.2.3	Nmfaf_3daDataManagement_Deconfigure service operation
main	4.2.2.3.1	General
main	4.2.2.3.2	Remove MFAF configuration
main	4.3	Nmfaf_3caDataManagement Service
main	4.3.1	Service Description
main	4.3.1.1	Overview
main	4.3.1.2	Service Architecture
main	4.3.1.3	Network Functions
main	4.3.1.3.1	Messaging Framework Adaptor Function (MFAF)
main	4.3.1.3.2	NF Service Consumers
main	4.3.2	Service Operations
main	4.3.2.1	Introduction
main	4.3.2.2	Nmfaf_3caDataManagement_Fetch service operation
main	4.3.2.2.1	General
main	4.3.2.2.2	Retrieve data or analytics from the MFAF
main	4.3.2.2A	Nmfaf_3caDataManagement_Subscribe service operation
main	4.3.2.3	Nmfaf_3caDataManagement_Notify service operation
main	4.3.2.3.1	General
main	4.3.2.3.2	Notification about the subscribed data or analytics
main	5.1	Nmfaf_3daDataManagement Service API
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: MFAF Configurations
main	5.1.3.2.1	Description
main	5.1.3.2.2	Resource Definition
main	5.1.3.2.3	Resource Standard Methods
main	5.1.3.2.3.1	POST
main	5.1.3.2.4	Resource Custom Operations
main	5.1.3.3	Resource: Individual MFAF Configuration

Document	Section	Title
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	PUT
main	5.1.3.3.3.2	DELETE
main	5.1.4	Custom Operations without associated resources
main	5.1.5	Notifications
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: MfafConfiguration
main	5.1.6.2.3	Type: MessageConfiguration
main	5.1.6.2.4	Type: MfafNotiInfo
main	5.1.6.2.5	Type: MfafTransferInfo
main	5.1.6.2.6	Type: TargetMfafNotiInfo
main	5.1.6.3	Simple data types and enumerations
main	5.1.6.3.1	Introduction
main	5.1.6.3.2	Simple data types
main	5.1.6.4	Data types describing alternative data types or combinations of data types
main	5.1.6.5	Binary data
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	5.2	Nmfaf_3caDataManagement Service API
main	5.2.1	Introduction
main	5.2.2	Usage of HTTP
main	5.2.2.1	General
main	5.2.2.2	HTTP standard headers
main	5.2.2.2.1	General
main	5.2.2.2.2	Content type
main	5.2.2.3	HTTP custom headers
main	5.2.3	Resources

Document	Section	Title
main	5.2.4	Custom Operations without associated resources
main	5.2.5	Notifications
main	5.2.5.1	General
main	5.2.5.2	MFAF Notification
main	5.2.5.2.1	Description
main	5.2.5.2.2	Target URI
main	5.2.5.2.3	Standard Methods
main	5.2.5.2.3.1	POST
main	5.2.5.3	Fetch Notification
main	5.2.5.3.1	Description
main	5.2.5.3.2	Target URI
main	5.2.5.3.3	Standard Methods
main	5.2.5.3.3.1	POST
main	5.2.6	Data Model
main	5.2.6.1	General
main	5.2.6.2	Structured data types
main	5.2.6.2.1	Introduction
main	5.2.6.2.2	Type: NmfafDataRetrievalNotification
main	5.2.6.2.3	Type: FetchInstruction
main	5.2.6.2.4	Type: NmfafDataAnaNotification
main	5.2.6.3	Simple data types and enumerations
main	5.2.6.3.1	Introduction
main	5.2.6.3.2	Simple data types
main	5.2.6.4	Data types describing alternative data types or combinations of data types
main	5.2.6.5	Binary data
main	5.2.7	Error Handling
main	5.2.7.1	General
main	5.2.7.2	Protocol Errors
main	5.2.7.3	Application Errors
main	5.2.8	Feature negotiation
main	5.2.9	Security

The purpose of the protocol as the source step located it in the documents: one DCCF configurator, MFAF, one analytics consumer; aggregated data has been collected, configuration → fetchInstruction notification → Fetch with fetchCorrIds → data response, a bounded transfer. The configuration procedure must be included; the nonexistent 3ca Subscribe operation must not be invented; upstream collection and complex processing are excluded.

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
A03	Notification and fetch correlation	the analytics consumer, assured that a notification it accepts carries the correId of its configuration, and the MFAF, assured that a Fetch it accepts carries fetchCorrIds of a fetchInstruction it really sent	the notification carries correId and the fetchInstruction (s0140-s0143, s0395); the Fetch carries the fetchCorrIds previously provided in that notification (s0120-s0121); the MFAF finds the data according to the requested parameters (s0122-s0124).	s0139, s0140, s0141, s0143, s0120, s0121, s0122, s0395, s0398
A08	Fetch uses its own notification's ids	the analytics consumer, assured that the fetchUri and fetchCorrIds it accepts come from the notification for its own configuration, and that the MFAF returns the data of the fetchCorrIds it requested and not another configuration's	fetchUri is assigned by the MFAF in the FetchInstruction (s0366, s0398) and the response is the stored data related to the fetch correlation identifiers (s0376, s0124).	s0366, s0398, s0374, s0376, s0124, s0120, s0121, s0395
Z01	Every operation carries an authorization	the MFAF, assured that each Configure, Notify and Fetch it performs or accepts is authorized by a token the NRF issued for that operation's service	access to both APIs may be authorized with OAuth2 client credentials, the NRF being the authorization server, and the consumer obtains a token before consuming the service (s0304-s0306, s0426-s0428); each NF shall validate all incoming messages (ts-33-501:2875).	main:1982, main:2634, ts-33-501:2875, s0304, s0305, s0306, s0426, s0427, s0428
Z03	No transfer of tokens or correlation ids	the analytics consumer and the MFAF: a consumer's token, correId and fetchCorrIds cannot be used for another consumer's configuration or for another operation	correId is chosen by the NF service consumer and the notification is identified by it alone (s0040, s0395); fetchCorrIds are strings the MFAF generates with no stated owner (s0398); the service has one scope and no per-resource scope (s0429); RFC 6749 says it provides no method for the resource server to ensure that a token presented by a client was issued to that client (rfc6749:2879).	main:2640, rfc6749:2879, s0040, s0395, s0398, s0121, s0429
Z04	Expired or revoked ids fetch nothing	the MFAF and the analytics consumer: after the configuration is deleted or the fetchInstruction expires, a Fetch with the old correId and fetchCorrIds returns no data	fetchInstruction carries an optional expiry as a deadline to fetch the data (s0398); Deconfigure removes the individual resource (s0070, s0075-s0076); the Fetch steps of the MFAF list no check of expiry or of the existence of the configuration (s0122-s0124).	s0398, s0070, s0075, s0076, s0122, s0123, s0124
S03	No stale data after reconfiguration	the analytics consumer and the MFAF: after an update of the configuration takes effect, an old fetchInstruction or old fetchCorrIds cannot return the data of the old configuration nor make the consumer accept old data as new	the update replaces the Individual MFAF Configuration identified by transRefId and stores it (s0059-s0065); the source defines no configuration version and no rule invalidating outstanding fetchInstructions on update (s0398).	s0059, s0060, s0061, s0062, s0064, s0065, s0398

ID	Name	About	What the documents say	Where
S04	Configuration, notification and fetch chained	the analytics consumer: the fetchCorrIds it uses come from the notification generated for its configuration, that notification's correId comes from that configuration, and no consumer or configuration is crossed with another	the consumer creates the configuration (s0035), the MFAF stores it (s0047-s0052), notifies with the correId of that configuration (s0140, s0395), the consumer stores the notification (s0148-s0149) and fetches with its ids (s0120-s0124).	s0035, s0047, s0052, s0120, s0121, s0122, s0124, s0140, s0148, s0149, s0395
R02	Notification accepted at most once	the analytics consumer: one logical MFAF notification is accepted at most once	the notification is identified only by correId and carries no message identity or sequence (s0395); the consumer stores the notification on acceptance (s0148-s0149); SBI signalling shall support replay protection (ts-33-501:2865).	ts-33-501:2865, s0140, s0141, s0395, s0148, s0149
R03	Fetch performed at most once	the MFAF and the consumer: one logical fetch operation has at most one security-relevant effect; a retransmission or a cached response is allowed	the Fetch carries only the fetch correlation identifiers (s0374) and the MFAF finds the stored data for them (s0122-s0124); the source states no side effect and no one-time use of the identifiers.	s0115, s0122, s0124, s0374, s0376
K02	Data visible only to its consumer	the analytics data the MFAF holds: the anaNotifications, dataNotif and dataReports of a fetchCorrIds are visible only to the consumer named in the configuration, not to other legitimate NFs nor to the external attacker	the notification and the fetch response carry the data to the configured notificationURI and to the requester (s0140, s0144-s0147, s0376); notifications shall comply with the SBI notification requirements (s0339) and SBI communication is confidentiality protected (ts-33-501:2915).	ts-33-501:2915, s0124, s0125, s0126, s0127, s0140, s0144, s0145, s0146, s0147
A08.11	Ids bound to the notified consumer	the analytics consumer, assured that the correId and fetchCorrIds in a notification correspond to the notificationURI and correId the configuration names for it, and that the MFAF accepts a Fetch only for the notification it made to that consumer	the notification is sent to the notificationURI of the configuration with the correId of the MessageConfiguration (s0140, s0395, s0268); the consumer stores the notification (s0148-s0149).	s0039, s0040, s0140, s0141, s0395, s0398
Z01.12	Fetch requires an authorization	the MFAF, assured that a Fetch is performed only for a party holding a token issued for the fetch service, so that identity authentication alone does not retrieve data	the Fetch is part of the Nmfaf_3caDataManagement API, whose access may be authorized with OAuth2 (s0426-s0427); the API defines a single scope and no resource or operation level scopes (s0429).	main:2634, main:2640, s0426, s0427, s0429
S03.13	Old instruction cannot outlive update	the analytics consumer and the MFAF: after a configuration update takes effect, an old fetchInstruction or old fetchCorrIds cannot make the MFAF return the data of the old configuration, nor make the consumer accept old data as the data of the new configuration	the update replaces the Individual MFAF Configuration identified by transRefId and stores it (s0059-s0062), answered with 200 OK or 204 No Content (s0064-s0065); the source defines no configuration version or epoch and no rule invalidating outstanding fetchInstructions on update (s0398).	s0059, s0060, s0061, s0062, s0064, s0065, s0398

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: 16. Not taken: 382. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

Paragraphs	Where	Taken up as	Why not
P001 to P002	rfc6749:2778-2786	No goal stated	the introduction to the OAuth2 security considerations points to a separate threat model; it states no requirement on this protocol.
P003 to P006	rfc6749:2790-2823	Outside the scope	client authentication and the confidentiality of client credentials belong to the OAuth2 baseline (TS 33.501, TS 29.510) which the task fixes and does not ask to analyse; the model takes token issuance and TLS client authentication from it.
P007	rfc6749:2827-2829	Z01, Z03, `rogue_nf`	
P008 to P010	rfc6749:2831-2850	Outside the scope	measures on the authorization server and on resource owner interaction, outside the flow analysed.
P011	rfc6749:2859-2865	Outside the scope	confidentiality of access tokens in transit and storage is the OAuth2 and TLS baseline, not a property of this notification and fetch flow.
P012	rfc6749:2867-2868	Outside the scope	the implicit grant is not used by this API; the client credentials grant is.
P013	rfc6749:2870-2872	Z01	
P014	rfc6749:2874-2877	Z03, Z01.12	
P015	rfc6749:2879-2881	Z03, A08, `rogue_nf`, `ids_disclosed`	
P016 to P033	rfc6749:2885-3003	Outside the scope	refresh tokens, authorization codes, redirection URI manipulation, resource owner password credentials: grant types and flows this API does not use.
P034 to P036	rfc6749:3013-3028	Outside the scope	request confidentiality and endpoint authenticity are properties of the OAuth2 and TLS baseline fixed by the task.
P037 to P039	rfc6749:3032-3041	Outside the scope	credential guessing concerns passwords; this flow uses client credentials and tokens, and no low-entropy secret is chosen in it.
P040 to P062	rfc6749:3045-3210	Outside the scope	phishing, cross-site request forgery, clickjacking, code injection, open redirectors and implicit grant impersonation are web client threats outside this NF-to-NF API flow.
P063	ts-33-501:2491-2491	No goal stated	bidding down attacks on algorithm negotiation; this flow negotiates no algorithm or protection level, so the paragraph states nothing that can be attacked here.
P064 to P072	ts-33-501:2495-2511	Outside the scope	authentication and authorization between UE and network in 5G AKA, not this API flow.
P073 to P212	ts-33-501:2515-2851	Outside the scope	keys, user data and signalling protection, subscription credentials, subscriber privacy, gNB and ng-eNB requirements: the access network and subscription part of 5G security, which the task excludes.
P213	ts-33-501:2853-2853	Outside the scope	requirements on the UDR, a data store outside the analysed flow.
P214	ts-33-501:2859-2859	Outside the scope	trust zone subdivision; the model assumes the DCCF, MFAF and consumer are in trust zones that reach each other only over the SBI.

Paragraphs	Where	Taken up as	Why not
P215	ts-33-501:2865-2865	R02, R03	
P216	ts-33-501:2867-2867	Z01	
P217	ts-33-501:2869-2869	Outside the scope	topology hiding between trust domains, not a property of this flow.
P218	ts-33-501:2871-2871	A03, Z01	
P219	ts-33-501:2873-2873	A03, Z03, `ids_disclosed`	
P220	ts-33-501:2875-2875	Z01, A03	
P221 to P222	ts-33-501:2879-2881	Outside the scope	NF profile and certificate matching and NF discovery, outside the analysed operations.
P223 to P224	ts-33-501:2883-2885	Z01	
P225 to P227	ts-33-501:2887-2891	Z01, Z03, `rogue_nf`	
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF exposure to application functions, not this flow.
P235 to P237	ts-33-501:2911-2915	Outside the scope	mutual authentication and protection between SCP and NFs is the SBI transport baseline fixed by the task.
P238	ts-33-501:2917-2917	K02, R02	
P239 to P240	ts-33-501:2919-2921	Outside the scope	colocation of SCP endpoints and protection of SCP internal interfaces, deployment matters outside this flow.
P241	ts-33-501:2923-2923	`ids_disclosed`	
P242 to P303	ts-33-501:2929-3055	Outside the scope	roaming interconnection, SEPP and N32-f requirements, outside the intra-PLMN flow analysed.
P304 to P312	ts-33-501:3059-3075	Outside the scope	protection of attributes over the N32-f interface between SEPPs, outside this flow.
P313 to P318	ts-33-501:3079-3091	Outside the scope	IPUPS and the originating network identifier header, outside this flow.
P319 to P337	ts-33-501:3097-3141	Outside the scope	signalling monitoring and security visibility and configurability, not properties of this API flow.
P338 to P375	ts-33-501:3149-3231	No goal stated	ciphering and integrity algorithm identifier values and algorithm selection for RAN and NAS: this flow selects no algorithm, so the paragraphs state nothing that can be attacked here.
P376 to P398	ts-33-501:6540-11901	Outside the scope	N3, EAP-TLS subscription identifier privacy, IAB and MBSF requirements, outside the analysed flow.

The catalogue of properties, entry by entry

The properties were drafted from a catalogue of security and privacy properties. The draft left out 32 of its entries, and the analyst went through each of them: examined here as a property, or left out with the reason. Examined: 10. Left out: 22. That the documents do not speak of a property is not a reason the analyst may give.

Entry	Property of the catalogue	Examined as	Why not
K01	Key/credential confidentiality	Outside the scope	the only keys and credentials of this flow are the TLS session and the OAuth2 access token, whose protection the task fixes as the TS 29.500, TS 33.501 and TS 29.510 baseline; the flow itself defines no key or credential.
K03	Strong secrecy/indistinguishability of secret values	Outside the scope	strong secrecy and indistinguishability are the equivalence route; the documents claim no indistinguishability of any value, and the confidentiality claim of this flow is asked as K02.
K04	Key separation by purpose, direction and session	Not applicable	the flow defines no key hierarchy, direction keys or session key derivation.
D01	Message integrity and data origin authenticity	A03, A08	
A01	Peer participation (aliveness)	A03	
A02	Weak agreement authentication	A03	
A04	Injective agreement authentication	R02, R03	
A05	Implicit key authentication	Not applicable	the flow establishes no session key, so implicit key authentication has nothing to speak of.
A06	Explicit key confirmation	Not applicable	the flow establishes no session key, so key confirmation has nothing to speak of.
A07	Identity–credential–key binding	Z03	
R01	Authentication/challenge freshness	R02	
R04	Message acceptance order/window security	Not applicable	the flow defines no sequence number, acceptance window or ordering rule.
R05	Resistance to downgrade/security policy preservation	Not applicable	the flow negotiates no version, suite or protection level that could be lowered.
C01	Forward secrecy (FS)	Not applicable	no session key is established, so forward secrecy has nothing to speak of.
C02	Resistance to key compromise impersonation (KCI)	Not applicable	no session key is established, so key compromise impersonation has nothing to speak of.
C03	Known-key security/leakage isolation	Not applicable	the flow defines no key derivation between sessions, so leakage isolation has no content.
C04	Ephemeral secrets/resilience to partial state leakage	Not applicable	the flow defines no ephemeral secret or partial state component.
C05	Post-compromise security (PCS)	Not applicable	the flow defines no ratchet or rekeying, so post-compromise security has no content.
C06	Enhanced authentication after registration credential compromise	Not applicable	the flow uses no registration record or password authenticated key exchange.
C07	Resistance to offline dictionary attacks	Not applicable	the flow uses no low-entropy password; the credentials are client credentials and tokens.
C08	Goal preservation under failure of a specified primitive	Not applicable	the flow claims no protection under the failure of a specific primitive, and the task fixes the primitives as the TLS and OAuth2 baseline.

Entry	Property of the catalogue	Examined as	Why not
Z02	No expansion of delegated authority	A08	
Z05	Cross-tenant/cross-service security isolation	Z03	
S01	Encryption nonce uniqueness	Not applicable	the flow specifies no nonce for an encryption mode.
S02	Retransmission/key reinstallation security	R02, R03	
S05	Termination and invalidation of old state	Z04	
P01	Identity anonymity	Outside the scope	identity anonymity is the equivalence route; the documents claim no anonymity, and the parties are identified NFs.
P02	Session unlinkability	Outside the scope	unlinkability of sessions is the equivalence route; the documents claim none, and correId links the steps by design.
P03	User/credential existence hiding	Outside the scope	hiding whether a user or credential exists is the equivalence route and is not claimed by the documents.
V01	Conditional progress/error recovery	The prover cannot be asked	conditional progress and recovery are liveness properties, which the prover cannot be asked.
V02	No unexpected deadlock	The prover cannot be asked	absence of deadlock is a liveness property, which the prover cannot be asked.
V03	Bounded resource abuse/resistance to amplification	The prover cannot be asked	resource abuse and amplification are cost properties, which the prover cannot be asked.

The parties and the protected values, one by one

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

ID	Party or value	Kind	Attacker	Why none
S01	DCCF	party	Everything lost by definition	the configurer decides the notificationURI and the correId of each configuration it creates; if it is dishonest, the data of its own configurations goes where it says, and the documents ask no property of this flow against a dishonest configurer.
S02	MFAF	party	Everything lost by definition	if the attacker controls the MFAF, it holds the data and the fetchCorrIds themselves; every property of this flow about the MFAF is lost by definition.
S03	analytics consumer	party	`ids_disclosed`	
S04	NRF	party	Everything lost by definition	a compromised authorization server can issue a token for any subject and scope, so Z01 and Z01.12 are lost by definition and the documents ask no other property against it.
S05	other legitimate NFs	party	`rogue_nf`	
S06	correId	value	`ids_disclosed`	
S07	fetchCorrIds	value	`ids_disclosed`	
S08	fetchUri	value	`ids_disclosed`	

ID	Party or value	Kind	Attacker	Why none
S09	notificationURI	value	Nothing gained	the notificationURI is an endpoint address that the configuration names; having it gives the attacker nothing beyond the network that the SBI does not already give it.
S10	transRefId	value	`ids_disclosed`	
S11	OAuth2 access token	value	Everything lost by definition	a party holding an NF's access token acts as that NF, so the authorization claims are lost for it by definition; the documents ask them only against a party without the token.
S12	service scope nmfaf-3dadatamanagement	value	`rogue_nf`	
S13	service scope nmfaf-3cadatamanagement	value	`rogue_nf`	
S14	expiry	value	Nothing gained	the expiry is a deadline carried in the fetchInstruction; holding it gives the attacker nothing beyond the network except knowledge of the same notification, which is covered by ids_disclosed.
S15	configuration version	value	Nothing gained	the source defines no configuration version; the values the update and deconfiguration claims rest on are covered by ids_disclosed.
S16	notification message instance	value	`ids_disclosed`	
S17	Fetch request instance	value	`ids_disclosed`	
S18	data fetch operation instance	value	`ids_disclosed`	
S19	anaNotifications	value	`ids_disclosed`	
S20	dataNotif	value	`ids_disclosed`	
S21	dataReports	value	`ids_disclosed`	
S22	the data corresponding to fetchCorrIds	value	`ids_disclosed`	

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		
`rogue_nf`	another legitimate NF in the same network: its own TLS credentials and an access token issued by the NRF for the service-wide scope nmfaf-3cadatamanagement (and nmfaf-3dadatamanagement), but no fetchCorrIds of other consumers	TS 29.522 defines one scope for the whole of each service and no resource or operation level scopes (main:1988, main:2640); the NRF issues tokens to NF service consumers (ts-33-501:2883-2885); the task boundary asks for the malicious legitimate NF with its own credentials.	main:2640, s0307, s0429, s0304, s0426

Attacker	Has	Basis	Where
`ids_disclosed`	the correId, fetchUri and fetchCorrIds of one notification, obtained outside the fetch operation: from an SCP that handles or redirects the notification, or from a previous or stale holder of the values	the SCP handles the messages it forwards and every party must trust it to handle them correctly (ts-33-501:2915-2921), and there is no mutual authentication between consumer and producer when an SCP is used (ts-33-501:2873); TS 29.522 gives no rule on the confidentiality, uniqueness or one-time use of fetchCorrIds and fetchUri, and none forbids a second NF from presenting them (s0398, s0120-s0123).	ts-33-501:2921, s0398, s0121, s0122, s0123, s0355, s0376

ID	`network`	`rogue_nf`	`ids_disclosed`
A03	Holds		Fails
A08	Holds		Fails
Z01	Holds	Holds	
Z03	Holds	Holds	Fails
Z04	Fails		Fails
S03	Fails		Fails
S04	Holds		Fails
R02	Fails		
R03	Holds		
K02	Holds	Holds	Fails
A08.11	Holds		Fails
Z01.12	Holds	Holds	
S03.13	Fails		Fails

Source and targets

Source: 477 units, 195817 bytes, frozen in source-delivery-01.json. Targets: 10 approved (A03, A08, Z01, Z03, Z04, S03, S04, R02, R03, K02); 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 bundle protocol-86e6555558a8460ca7956850a3e1

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 101532 tokens in and 5580 out, 265 s.

Notes on the prover and the model

- Z04: answered in ProVerif, and the analyst names changing_state for this target
- S03: answered in ProVerif, and the analyst names changing_state for this target
- S03.13: answered in ProVerif, and the analyst names changing_state for this target

For a ProVerif model that simplifies algebra (xor, dh_algebra, other_algebra), the attacker is weaker than the real operation: a proof there is a proof against a weaker attacker, an attack there is an attack. For a ProVerif model that leaves out a check on a value that changes (changing_state), parties accept more: a proof there holds, an attack there may be nothing but the missing check. Tamarin+ models these as they are.

State review by the analyst

This part is the analyst's own reading of the source, not an answer of the prover. "yes" under shown by ProVerif means that ProVerif found a trace with these steps in the unrolled model, so the steps are possible; that they break a target is said only by an attack in the table of answers, and what else they cost is the analyst's reading. "nothing found" says which patterns were looked at and proves nothing.

value	kept by	changes when	checked when	refs
the Individual MFAF Configuration: transRefId, correId, notificationURI	MFAF	a Configure POST creates it (s0047-s0052), a PUT on /configurations/{transRefId} replaces it (s0059-s0062), a DELETE removes it (s0075-s0076)	the MFAF reads it to address the notification to its notificationURI and to set the correId of the notification (s0140, s0395); on update and delete it is identified only by the transRefId in the request URI	s0047, s0048, s0049, s0050, s0051, s0052, s0059, s0060, s0061, s0062, s0075, s0076
the fetchCorrIds of a notification, with the expiry deadline of the fetchInstruction	MFAF	the MFAF generates them for a fetchInstruction (s0398) and they should become void when the fetchInstruction's expiry passes (s0398) or the configuration is removed (s0076)	on every Fetch: the MFAF looks the data up by the identifiers the requester carries (s0122-s0124, s0374)	s0398, s0120, s0121, s0122, s0123, s0124, s0374, s0376
the stored fetchInstruction: correId, fetchUri, fetchCorrIds	analytics consumer	the MFAF notifies it and the consumer stores the notification (s0148-s0149); nothing tells it when the configuration behind the values is updated or removed	when the data or analytics arrive in the fetch response: the consumer can compare the correId of the response with the correId of its stored fetchInstruction (s0395)	s0140, s0148, s0149, s0153, s0395

pattern	result	target	shown by ProVerif	steps
wrap_or_window	candidate	Z04	yes: ProVerif falsifies `z04_expiry` on bounded.pv	MFAF creates the configuration and sends a fetchInstruction with fetchCorrIds and an expiry deadline (s0140, s0398). / The consumer fetches inside the deadline; the MFAF answers 200 with the data (s0122-s0124). / The same consumer fetches the same identifiers again after the deadline; the MFAF's steps compare nothing with the deadline, so it answers 200 with the same data again.
desynchronisation	candidate	S03.13	yes: ProVerif falsifies `s03_accept` on bounded.pv	The DCCF creates the configuration with correId A; the MFAF notifies the consumer with correId A and fetchCorrIds I, and the consumer stores them (s0140, s0148-s0149). / The DCCF updates the same configuration by PUT /configurations/{transRefId} with new contents; the MFAF stores the new configuration (s0059-s0062). / The consumer has no new notification and keeps A and I; it fetches with I, the MFAF serves the old configuration's data with correId A, and the consumer accepts it as the data of the configuration it knows.
replay	candidate	R02	no: not tried with ProVerif	The MFAF sends the notification for correId A with fetchCorrIds I. / The consumer accepts it, stores it and answers 204 (s0148-s0150). / The same notification is delivered again, for instance by an SCP retry; nothing in the source identifies a notification instance, so the consumer accepts and stores it a second time.
update_before_check	candidate		no: not tried with ProVerif	An NF with the configuration service scope learns a transRefId, for instance from a Location header it once saw or from a log. / It sends PUT /nmfaf-3dadatamanagement/<apiVersion>/configurations/{transRefId} with its own MfafConfiguration (s0059). / The MFAF updates and stores that configuration without comparing the requester with the configuration's owner, so the notificationURI and correId of another consumer are replaced.

pattern	result	target	shown by ProVerif	steps
race_between_sessions	candidate		no: not tried with ProVerif	The MFAF holds the data of fetchCorrIds I for configuration A. / Two parties, or two copies of the consumer, send a Fetch with I at the same time. / Both are served with the same data, and nothing in the source says the second must be refused.
distinguishable_failure	candidate		no: not tried with ProVerif	A fetch with unknown identifiers gets an error response (s0417); a fetch with identifiers of a deleted configuration is served (s0122-s0124 in the model). / The difference tells the sender whether data for a given set of identifiers still exists, which is information about another consumer's configuration.
repeat_one_time_action	nothing found			Nothing in the source states a value that is installed once and must not be installed again, and the stored configuration is only created, replaced or removed by the operations the source defines.
no_rollback_on_failure	nothing found			A Fetch that fails receives an error response (s0417) and changes no stored value, and a refused Configure, update or Deconfigure leaves the configuration as it was; the source states no rollback step and none is needed.

What the analyst named for targets:

step	targets	features	reason
29	Z04, S03, S03.13	changing_state	These three targets rest on state that the MFAF keeps and changes: the configuration created, updated by PUT and removed by DELETE, and the lifetime of the fetchCorrIds the MFAF handed out, bounded by the optional expiry of the fetchInstruction. ProVerif cannot follow such a value for any number of sessions, so model.pv leaves the update, the deletion and the deadline out; bounded.pv unrolls one configuration through create, notify, fetch, update, fetch, deconfigure, fetch. Tamarin+ Agent, which models the configuration store and the deadline as they are, would settle what lies beyond the three fetches unrolled.
29	A03, A08, A08.11, S04, Z01, Z01.12, Z03, K02, R02, R03	plain_primitives, many_sessions	Every other target rests only on symmetric encryption, a signature, a MAC-free token check and a pairwise TLS key, with no algebraic law beyond the ideal primitives, asked over any number of sessions of the three roles. The answers rest on the endpoint TLS abstraction (one authenticated encrypted channel per pair of NFs) and on the OAuth2 token being a signature the authorization server alone can make.

The analyst's closing note

Thirteen targets answered on a ProVerif model of TS 29.522 configured notification and fetch (model.pv: DCCF, MFAF, analytics consumer, NRF tokens, a second legitimate NF and a disclosing intermediary; bounded.pv: the configuration lifetime over three fetch rounds). Proved: the notification and fetch correspondence (A03, A08, A08.11), the authorization of every Configure and Fetch by an NRF token for the operation's scope, with an outsider holding only another service's token refused (Z01, Z01.12), at-most-one effect of a fetch (R03), and the secrecy of the data from the network attacker (K02). Falsified, each with its attack recorded and drawn: a second NF holding a notification's fetchCorrIds is served that consumer's data with any token for the service-wide scope (A03, A08, A08.11, Z03, K02, S04); the expiry of the fetchInstruction and Deconfigure revoke nothing, so old identifiers keep returning the data (Z04); an update never reaches the consumer's fetchInstruction, so pre-update data comes back and is accepted as current (S03, S03.13); a duplicated notification is stored twice, reported as an ambiguity because the model's TLS abstraction lacks the SBI replay protection the source requires (R02). Four findings are reported, three of them weaknesses of the specification as written with the exact attacker they need, and none of the three recommendations was checked with check_fix within the session budget, which each finding says. Blocking: the

ownership gap depends on whether a notification's identifiers can reach a third party, which the source does not settle and which Tamarin+ Agent could model with the SCP path; the update/delete of a configuration by transRefId with no ownership check was found in the state review but not confirmed by a query, so it is a limit and not a finding.