OneDro1d Argus

CASE STUDY

Six differences between two releases of a ticketing system

One sealed set of checks, run with OneDroid Argus compare against OTRS Community Edition 6.0.30 and its fork Znuny 7.3.7: what changed, what stayed the same, and what the test could not see.

Case study. Runs of 5 and 6 October 2026. OneDroid Argus compare, in early use.

In short

- An engineer on our team and his coding agent took two releases of an open source ticketing system and ran one sealed set of checks against both. The people who build OneDroid Argus did not choose the system and did not write a check.
- The wider of two sets had 26 checks and ran three times on each system. OneDroid Argus reported 23 cells identical, 9 that differ, none noisy and none not measured.
- Six differences between the two releases. In one of them the older release returns an emoji as a replacement character, where the newer one returns it unchanged.
- Error paths, customer permissions, the ticket workflow, search and the response time of the sign-in page stayed the same.
- The test is narrow: one REST web service and four pages, fresh installs, one laptop. The limits are listed below in full.

The question

A team that moves from an old release to a new one, or to a fork, wants to know what changes for everything that talks to the system. A change log answers with what its authors remembered to write down. A comparison answers by running the same requests against both and reading what comes back.

The reference was OTRS Community Edition 6.0.30. The candidate was Znuny 7.3.7, the open source fork of it. This is not a review of either product and not a recommendation. It is a record of what one set of checks saw.

The setup

Item	Value
Reference	OTRS Community Edition 6.0.30
Candidate	Znuny 7.3.7
How they were built	Both from the official release archives with pinned checksums, on the same base image
Database	MariaDB 10.11 for both
Starting data	The same fixture for both
Host	One Windows laptop, Docker Compose, two OneDroid Argus executors side by side, no cluster
OneDroid Argus	Compare, in early use, with our hosted development control plane
Runs	Three per system, per set
Who wrote and ran the checks	One engineer's coding agent

What was checked

Set 1, 5 October, 16 checks. A seed check. Ticket create, update, history and search over the REST web service. Five error and permission paths. Session data. Three entry pages. The old URL path. One load check.

Set 2, 6 October, 26 checks. The same ground, plus: a pending reminder, close, a queue move with owner and lock, a customer refused a ticket that is not theirs, a customer raising the priority of their own ticket, an update to a state that does not exist, an emoji in an article, an article with an attachment, a read with a customer session, and the session check again with its values masked.

One set holds three kinds of check. A **measured** check records the output of each system and compares the two, field by field. A **fixed** check states a claim that every system must meet by itself, the reference included. A **load** check says the candidate must not be worse than the reference by more than a declared band, here 25%.

Before the comparison ran, each set was sealed and its hash was recorded on a ledger, so a set could not be changed after anyone had seen a result. One thing the seal does not show: the checks were rehearsed on both systems before sealing. The seal proves the set did not change afterwards. It does not make its author blind to the systems.

The result

	Set 1, 5 October	Set 2, 6 October
Checks	16	26
Runs per system	3	3
Cells identical	16	23
Cells that differ	4	9
Cells that read as noise	1	0
Not measured	0	0
Could not run	0	0
Verdict	differences	differences

The six runs of the first set took about 8 minutes. The numbers are the ones OneDroid Argus reported. The verdict is the word differences with the table behind it. OneDroid Argus gives no score.

The six differences

What	OTRS 6.0.30	Znuny 7.3.7	How it was seen
The old URL path `/otrs/index.pl`	Answers	Answers 404	Fixed check, 0 of 3 runs on Znuny
An emoji in an article	Comes back as U+FFFD, the replacement character	Comes back unchanged	Fixed check: 0 of 3 on OTRS, 3 of 3 on Znuny
`CommunicationChannel` on every article	Not there	There	Measured, on three different reads: plain, with an attachment, with a customer session
The `DynamicField` list on a ticket	The list, with empty values	No list	Measured. The cause was not found
Ticket history after create, close and reopen	11 entries	13 entries	Measured. The extra entries are `SetPendingTime`, written at a state change
Keys in the session data	1 key only here	7 keys only here	Measured with the values masked. 26 keys are common

We did not trace any of these into the source code, and we do not say which are intended. During the first set, a keyword search of the Znuny change log did not find the `CommunicationChannel` field,

the missing `DynamicField` list or the extra history entry. A keyword search is a weak test, so read that as not found by us, not as undocumented.

The emoji row is the one to look at twice. It is a fixed check, so the reference is judged like any other system, and the reference is the one that fails. A comparison that only asked whether the new release matches the old one would have counted the newer, correct behaviour as a regression.

Only the read was checked. The study does not say what OTRS 6.0.30 stores in its database for that article.

What was the same

- Six error and refusal paths answer the same on both. For the five in the first set that is the same status, code and message in 3 of 3 runs each.
- Customer permissions: a customer is refused a ticket that is not theirs, and can raise the priority of their own.
- Pending reminder, close, and a queue move with owner and lock.
- Ticket search and the entry pages.
- The sign-in page under five users at once: 95th percentile 49 ms on OTRS and 54 ms on Znuny in the first set, 60 ms and 59 ms in the second. Both inside the 25% band.

The timings come from one laptop, five users and one page. They say the candidate was not slower in this run. They are not a benchmark of either product.

What the noise taught us

In the first set the session-data check compared raw values. The reference gave three different outputs in three runs, so OneDroid Argus marked the cell as noise and did not call it a difference. In the second set the values were masked and only the keys were compared. The noise was gone and a real difference showed: one key only on OTRS, seven only on Znuny.

Two things follow. Three runs per system is what makes unstable output visible at all. And a check on anything with ids, tokens or timestamps in it needs its masks before it can say something.

What this test did not cover

Not covered	Why it matters
Anything beyond one REST web service and four pages	Most of both products was never asked a question
A migrated database	Both systems were fresh installs with the same starting data. An upgrade with years of real data is the common case, and it was not tested
Browser checks, database checks, mail, service level rules and access control lists	The sets held HTTP checks only. OneDroid Argus can drive a browser, but a browser check inside a comparison has not been tried
More than three runs per system	Three runs show unstable output. They do not show a rare failure
Performance	One laptop, five users, one page. A parity signal and nothing more
Tolerances for numbers	Not used in either set
The cause of each difference	None was traced into the source. For the `DynamicField` list the cause was looked for and not found
An independent tester	The same team wrote the checks, ran them and wrote this, and OneDroid Argus is our product. The independent check is to run the set again
A released product	Compare is in early use and ran here against a development control plane

What a comparison can and cannot measure

It can say	It cannot say
Whether the same request gives the same answer on both systems, field by field	Whether a difference is intended. A person decides that
Whether each system meets a fixed claim by itself	Anything that no check asks
Whether the candidate is slower than the reference by more than a declared band	How fast either system is in production
Whether an output is stable from one run to the next	How a system behaves on data the set did not create

What we take from it

- A comparison of two releases of a real system that keeps state ran end to end on one laptop, with sign-in, saved session ids and saved ticket ids carried between steps.

- The most useful finding came from a fixed check that the reference failed. The old system is not right by definition.
- Mask first, then compare. An unmasked check produced noise, the masked one produced a finding.
- Three of the six differences were not found by a keyword search of the change log. Whether they are documented elsewhere, we did not establish. Running both releases found them in minutes.

The video: https://www.youtube.com/watch?v=s_OzzMsZEXM

OTRS and Znuny are names that belong to their respective owners. OneDroid is not affiliated with either project, and the product names are used only to identify the software that was tested. OneDroid Argus is built by Providentia Worldwide. Questions: michal@onedroid.ai. How OneDroid Argus works: onedroid.ai/argus.