USER DOCUMENTATION

Coordinated Vulnerability Disclosure

A Practical Guide for CNAs, GNAs, CSIRTs, and Publishers

Vulnerability-Lookup contributors

Updated 2026-11-20

www.vulnerability-lookup.org

Contents

Preface 2

Acknowledgements 4

Getting Started 5

Roles and handoffs 6

Prepare your instance 8

Handling a Disclosure 10

Intake and triage 11

Investigation and coordination 16

Publishing and Follow-up 19

Write and evaluate a vulnerability record 20

Publish as a CNA 29

Publish as a GNA 38

Publish the advisory and follow up 49

Maintain a local KEV catalogue 52

Use Cases 65

Coordinate an upstream and downstream release 66

Draft a security advisory from a patch 67

Appendices 73

Case and release checklists 74

References 75

Preface

This guide shows how to use Vulnerability-Lookup to receive a vulnerability report, coordinate a response, publish an advisory, and keep public information current. It follows the handling lifecycle in GCVE BCP-02, version 2.0 and links to the product documentation for release-specific configuration and procedures.

The disclosure lifecycle

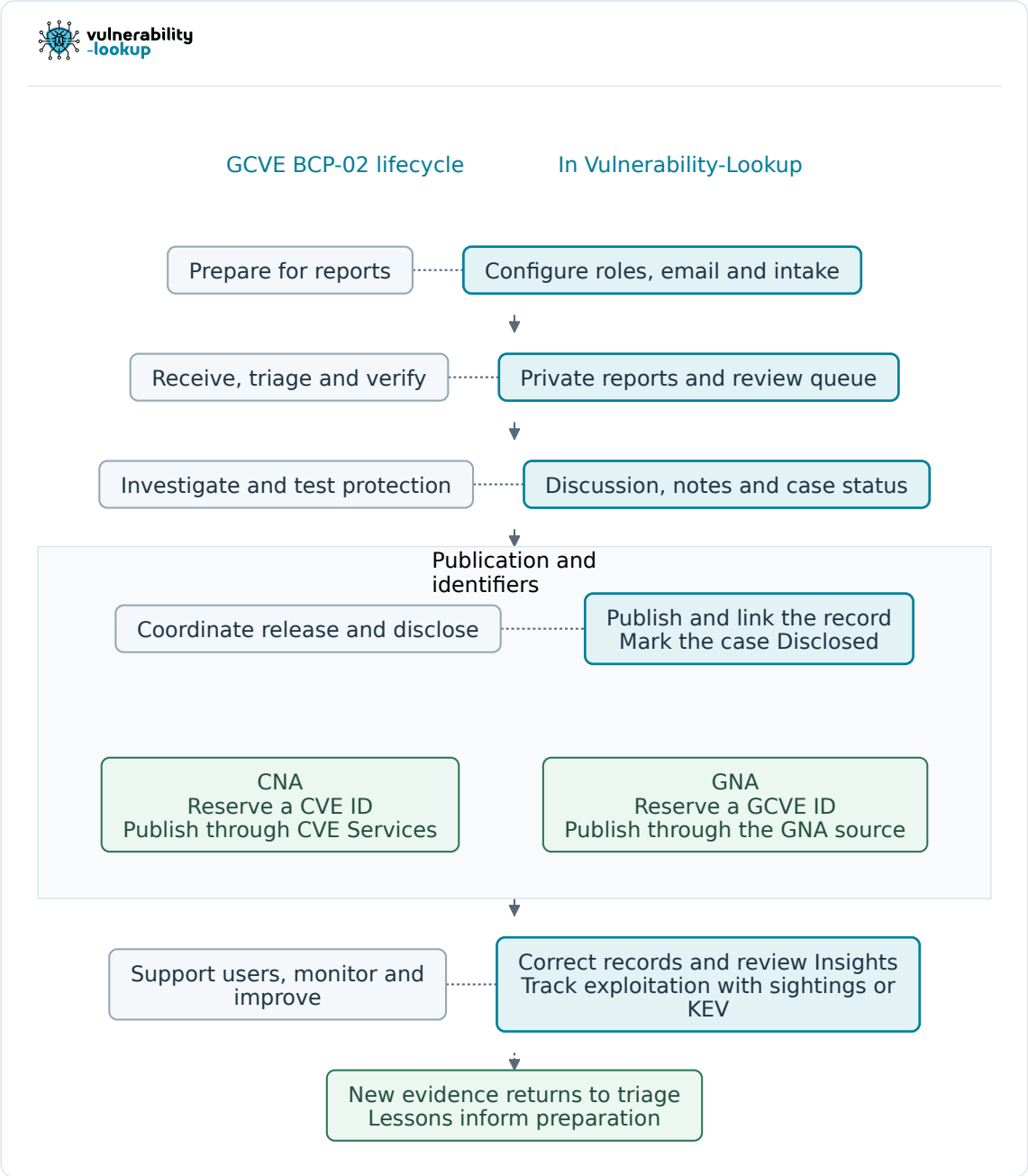


Figure 0.1: The GCVE BCP-02 lifecycle with Vulnerability-Lookup support and the CNA and GNA publication routes.

Keep the reporter and affected parties informed throughout. Maintainers develop, test, and deliver fixes through their engineering and release workflows. Where the Cyber Resilience Act applies, run regulatory reporting in parallel through the Single Reporting Platform. BCP-02 lifecycle overview · CRA reporting

When your team owns identifier assignment, use the agreed CNA or GNA route. Teams can also publish under local identifiers or coordinate with a partner that assigns the identifier. CNA publication · GNA publication · Local publication

Publication details

Editor: CIRCL Computer Incident Response Center Luxembourg

ISBN: 978-99987-458-1-0

Where to start

If you need to…	Start here
Set up a coordination team and a test instance	Roles and handoffs and Prepare your instance
Receive and coordinate a report as a CSIRT or maintainer	Intake and triage and Investigation and coordination
Reserve an identifier and publish a record	Publish as a CNA or Publish as a GNA
Release an advisory and verify that readers can find it	Publish the advisory and follow up
Rehearse the handoffs with multiple teams	Coordinate an upstream and downstream release

A worked Example Portal disclosure starts at Intake and triage. Follow the same case through the reporter and administrator views, coordination, approval, and signed-out release checks.

A case can span three records with different audiences. Keep confidential coordination in the disclosure case. Use the vulnerability record to carry the identifier and approved public data. Give users instructions for protecting their systems in the advisory. The release checklist brings these together.

Acknowledgements

The CIRCL team at the Computer Incident Response Center Luxembourg produced this book. It draws on the coordinated vulnerability disclosure (CVD) work of Alexandre Dulaunoy, Cédric Bonhomme, and David Cruciani.

We thank everyone who uses and contributes to Vulnerability-Lookup, including CSIRTs, fellow CNAs, and GNAs. Your feedback and contributions help improve how teams coordinate and publish disclosures.

We also thank the GCVE initiative for developing BCP-02, the practical guide that informs the disclosure process in this book. GCVE BCP-02

PART 01

Getting Started

Roles, handoffs and instance preparation.

CHAPTER 01

Roles and handoffs

Assign a case owner, a backup, and a release approver before reports arrive. One team can hold several roles. Record who owns the next action when responsibility passes between teams.

Role	Responsibility	Handoff
Reporter	Describe the issue, clarify affected scope, and state credit preferences.	Submit through the private reporting channel and keep the case reference.
CSIRT or other coordinator	Own intake, triage, reporter communication, and coordination across affected parties.	Send each maintainer the information needed to assess its products and agree on the next update.
Maintainer or product security team	Confirm affected releases, prepare and verify mitigations or fixes, and plan delivery.	Supply affected and fixed versions, remediation instructions, and a release decision.
CVE Numbering Authority (CNA)	Assign and publish CVE records within its authorized scope.	Return the CVE ID and verify the published CVE record.
GCVE Numbering Authority (GNA)	Assign identifiers within its GNA namespace and publish GCVE records.	Return the GCVE ID and verify the public record and its distribution.
Publisher	Review and release the user-facing advisory, then maintain corrections.	Give the coordinator the public advisory URL and confirmation of release.

1.1 Grant access to the disclosure workflow

Account	Access in Vulnerability-Lookup
Reporter	Needs `vulnerability_disclosure:report` to submit. The creation policy grants the reporter the scoped `vulnerability_disclosure_owner` role on the new case, allowing them to view it, join its discussion, and edit the report in the states that allow edits.
Disclosure manager	`vulnerability_disclosure:review` grants access to the case, discussion, private notes, and external reference. A resource-scoped assignment applies to one disclosure. A global assignment applies to every disclosure and is reserved for managers authorised to read them all.
Administrator	`admin:access` gives access to **Admin → Disclosures** and the **Review** form that sets state and disclosure date. The built-in admin role also has the disclosure management permissions.

1.1.1 Assign a manager to one disclosure

1. In **Admin → Roles**, create a role with `vulnerability_disclosure:review`. Clear **Role can be assigned globally** and select **Vulnerability Disclosure** under **Scoped resource types**.
2. In **Role assignments**, select the manager, the new role, and **Vulnerability Disclosure** as the **Scope**. Enter the numeric case ID in **Resource ID**. Find it at the end of the administrator's **Review** URL (/

`admin/disclosure/edit/<id>`). Select **Add assignment**, then share the private case link with the manager.

3. From the manager's account, check that the assigned case opens with its **Discussion**, **Notes**, and **External reference**. Try the link to a different private case with the same account and confirm access is denied. Check the account's other roles for global disclosure permissions if the second case opens.

An administrator handles the admin queue and state changes. Identifier reservation and release use the permissions for the chosen publication route. CVD roles · Resource-scoped assignments

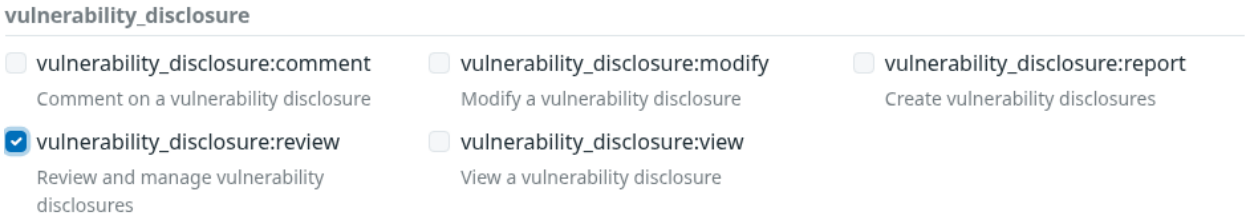


Figure 1.1: Select the review permission in the role editor. The role and its assignment determine which disclosures it covers.

BCP-02 · Roles and Responsibilities

CHAPTER 02

Prepare your instance

2.1 Configure report intake

1. Publish a disclosure policy with the products in scope, a secure contact channel, and the updates reporters can expect. Assign a team mailbox and a backup owner.
2. On a staging instance, set `user_accounts` to `true` in `config/generic.json`. In `config/website.py`, set `"vulnerability_disclosure": True` inside `WEB_MODULES`, and set `ADMIN_EMAIL` and `ADMIN_NAME` to the coordination mailbox and team name. Add `CVD_POLICY_TITLE` and `CVD_POLICY_URL` to show the policy on the report form. Use `CVD_POLICY_LOGIN_MESSAGE` for sign-in instructions. Apply any pending database migrations and restart the website. CVD configuration
3. Review the role assignments and your **Self-Registration Policy** before offering public registration. Sign in with a reporter account and open `/disclosure/new/`. Check that it can submit and that the displayed policy is the one you intend reporters to use.

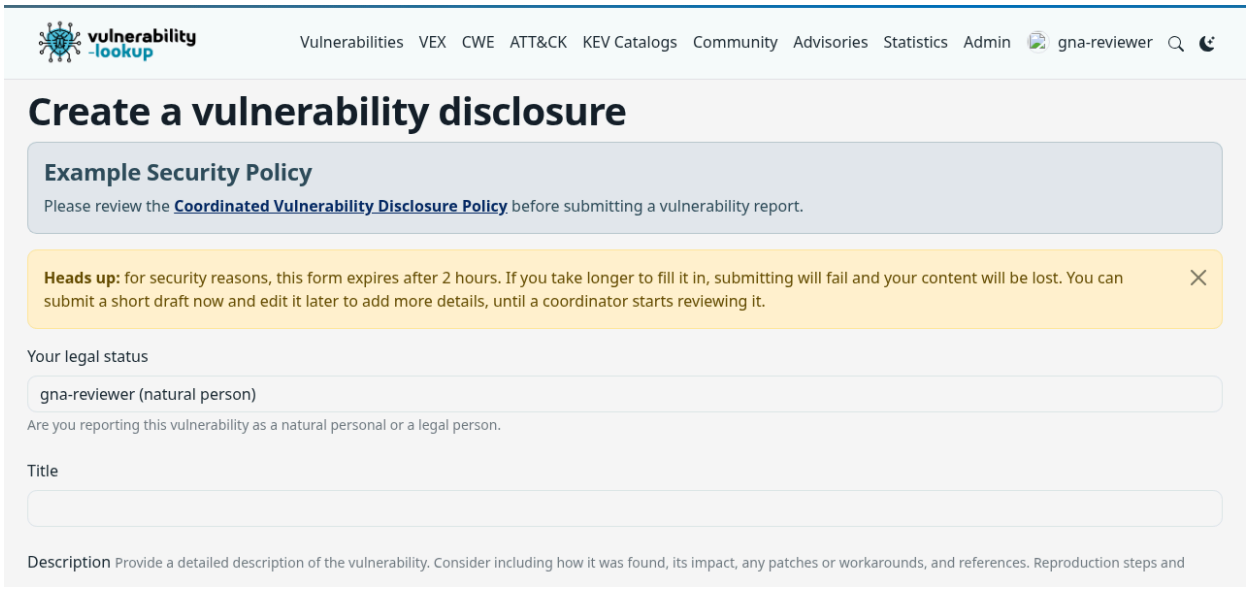


Figure 2.1: A configured disclosure policy appears at the top of the report form.

2.1.1 Rehearse notifications and access

Submit a fictional report and use separate reporter, manager, and administrator accounts to check access. Confirm time-sensitive notifications arrive at the mailbox that is responsible for the next action. CVD e-mail events

Action	Expected recipient
Reporter submits a new report, edits it, or posts a message	Coordination mailbox (`ADMIN_EMAIL`)
Team posts a message	Reporter
Administrator changes the state or disclosure date	Reporter, in one review message per save
Manager adds a private note or an administrator changes only the external reference	No e-mail

Check that an unrelated account cannot read the case. If you assign a manager to a single disclosure, use a second private case to verify the assignment's scope. Repeat the public check while signed out at **Approved**. A complete rehearsal then publishes the example and verifies which fields are visible. Intake and triage · Verify as a reader

2.2 Choose a publication route

Use a minimal publication deployment when the instance's main job is to publish records. Set up CNA publication for CVE Services or GNA publication for the allocated GCVE namespace. If another organization assigns the identifier, agree in advance who will reserve it, publish the record, and communicate any corrections.

BCP-02 · Preparing for Vulnerability Handling

PART 02

Handling a Disclosure

Intake, triage, investigation and coordination.

CHAPTER 03

Intake and triage

This walkthrough follows a fictional report about Example Portal from intake to public disclosure. The screenshots were taken on a local Vulnerability-Lookup 7.1.0 development instance. Vulnerability-Lookup revision b4c82c1

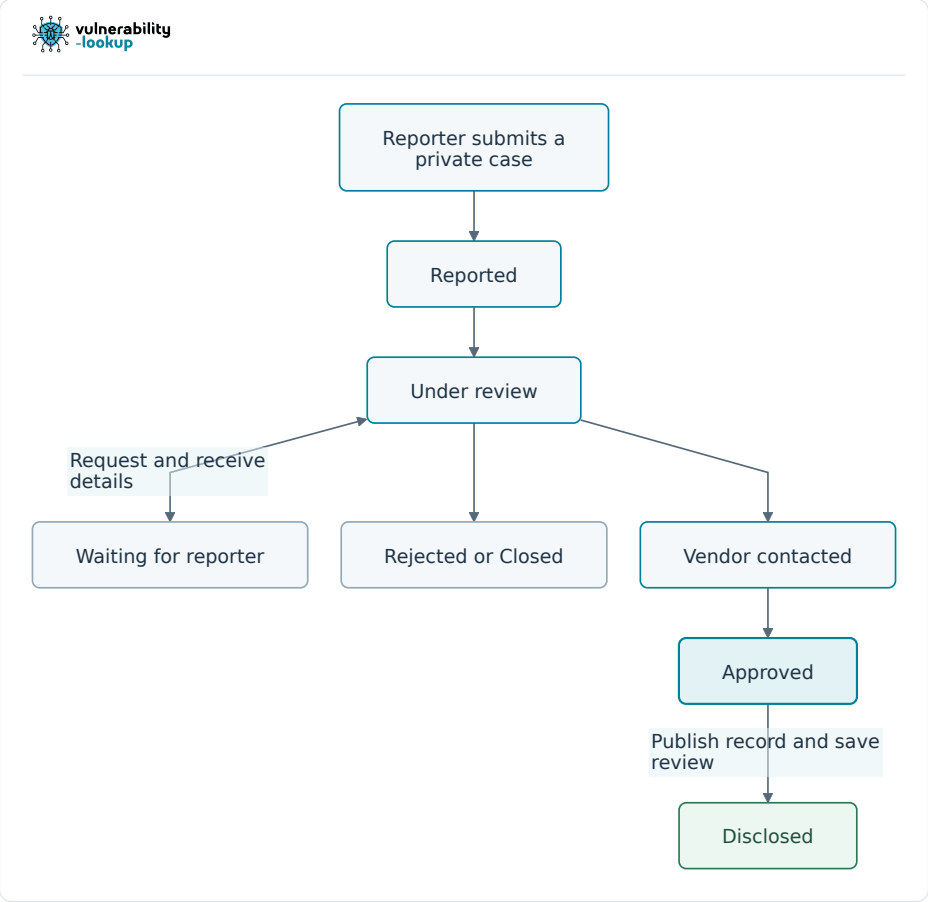


Figure 3.1: A report stays private during coordination and becomes public at the explicit release step.

3.1 Submit and follow a report

1. Sign in and open `/disclosure/new/`. Review the disclosure policy and choose whether you are reporting as an individual or on behalf of an organisation.
2. Enter a **Title** and a **Description** with the affected product or service, known impact, and any available mitigation. Draft the description for eventual publication. Keep sensitive material such as private evidence and identifying data in **Confidential details** or in the team's restricted case system.

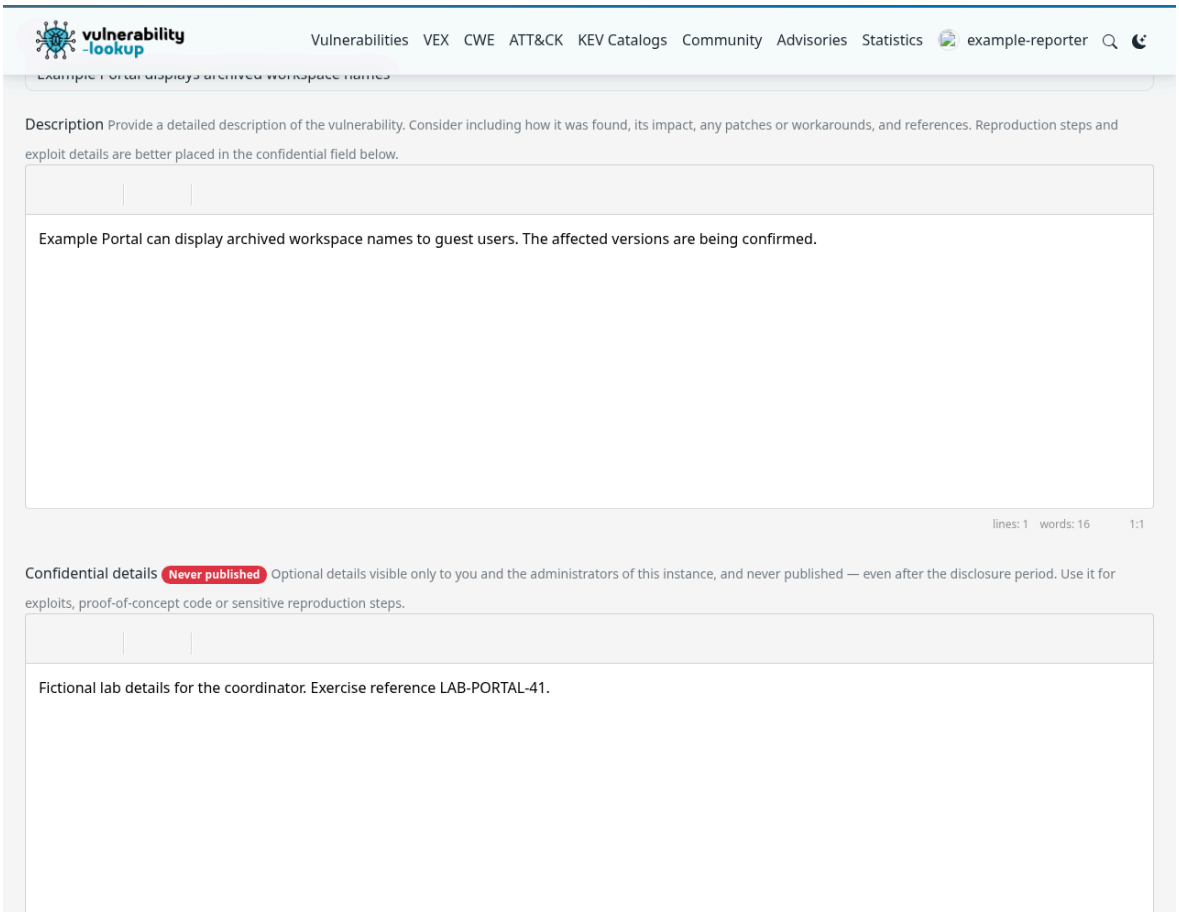


Figure 3.2: The report form separates the public-facing description from confidential details.

3. Select **Online Service** if the issue affects a service, or use **Affected Products** for installed software. Add known CWEs, CAPECs, or a **Vulnerability ID** if available. Record the credit preference or select **Remain Anonymous** for an uncredited report. Select **Submit**.

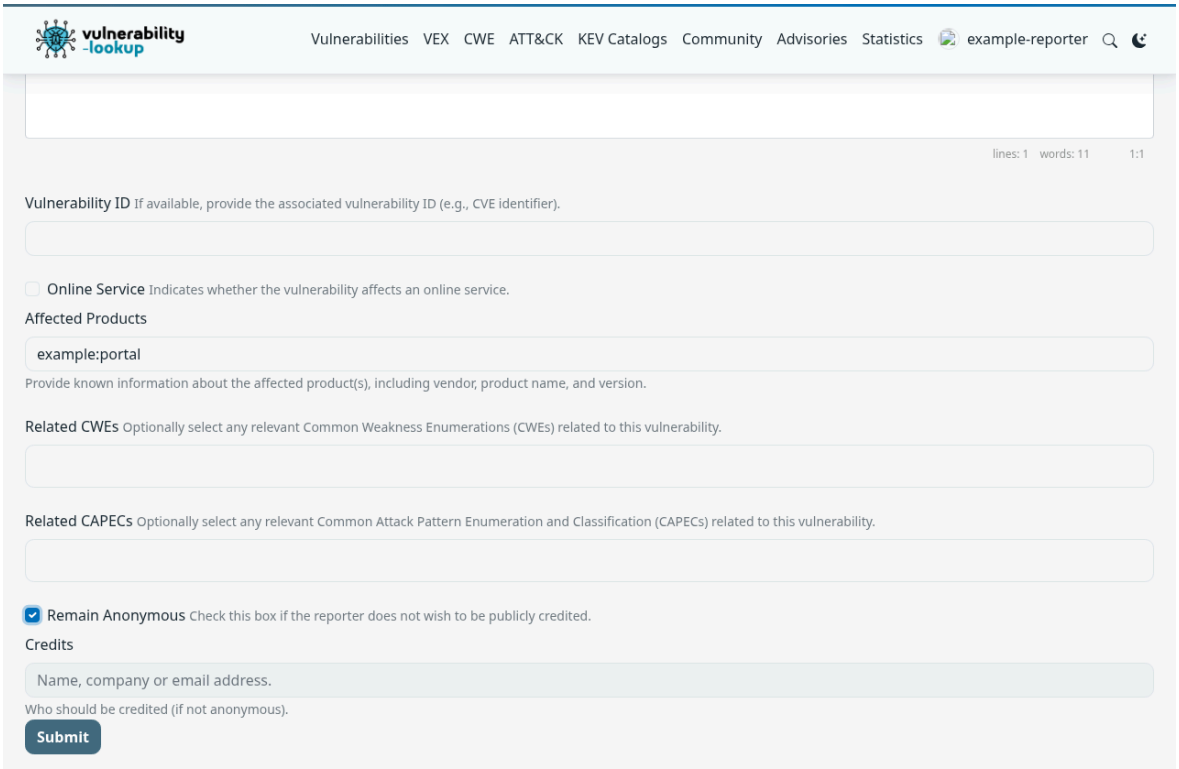


Figure 3.3: The reporter provides product scope and selects anonymous public disclosure.

4. Open **Your disclosures** from the case, or **Vulnerability disclosures** from the account menu. Open the report to check that it is **Reported**, and keep the private case link. **Overview** shows the state and progress. **Discussion** is where you exchange updates with the team. Reporter workflow

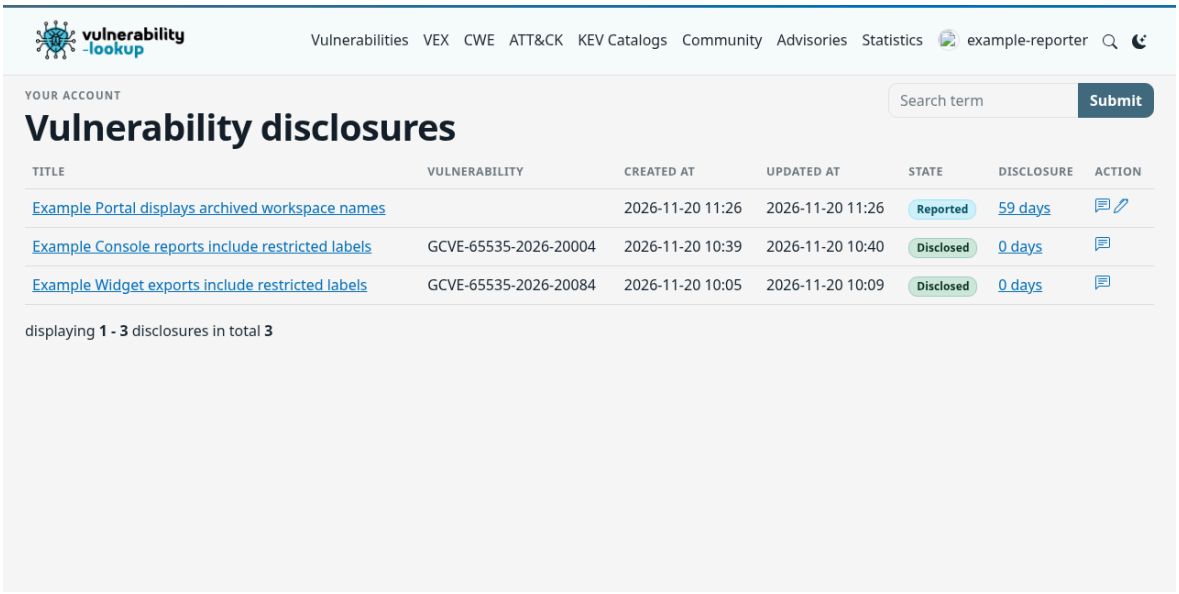


Figure 3.4: The reporter can see the new case and its state in their own list.

3.2 Acknowledge and triage

1. From **Admin → Disclosures** (`/admin/disclosures/`), select **Reported**. Search by title, description, reporter, or external reference when locating a specific case. Open the report and use **Overview** to check the description and any confidential details. The copy buttons on those cards copy the corresponding Markdown source.

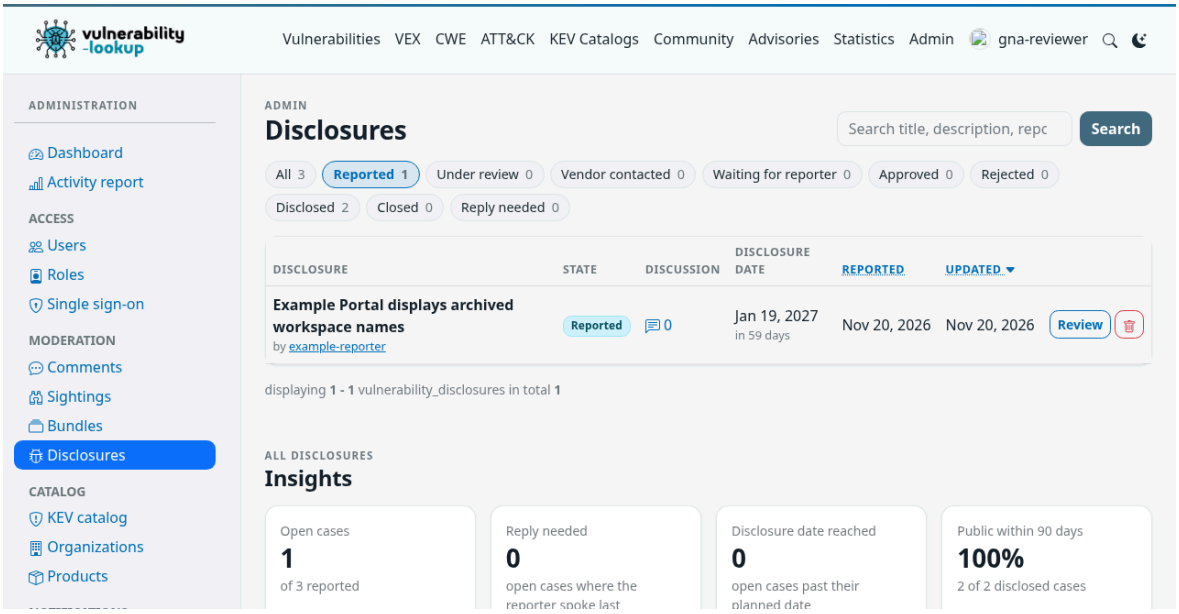


Figure 3.5: The administrator filters the queue to new reports and opens the case.

2. Assign an owner and record the next action in your team's case tracker or a private **Note**. Select **Review**, set **State** to **Under review**, and select **Save review**. In **Discussion**, acknowledge the report and give the reporter a date for their next update. Select **Post message** to send the update through the case.
3. Check that the product is in scope, look for duplicate reports or existing identifiers, and record what you can confirm about impact and versions. Use **Rejected** for a duplicate or out-of-scope report and **Closed** for a case ending without publication. Explain the decision to the reporter in **Discussion**.

3.3 Ask for missing information

Set the case to **Waiting for reporter** in **Review** and describe what you need in **Discussion**. The reporter can use **Edit report** only while the case is **Reported** or **Waiting for reporter**. In other states, they can send corrections in **Discussion**. A saved edit adds a history entry naming the fields that changed and sends a notification to the coordination mailbox.

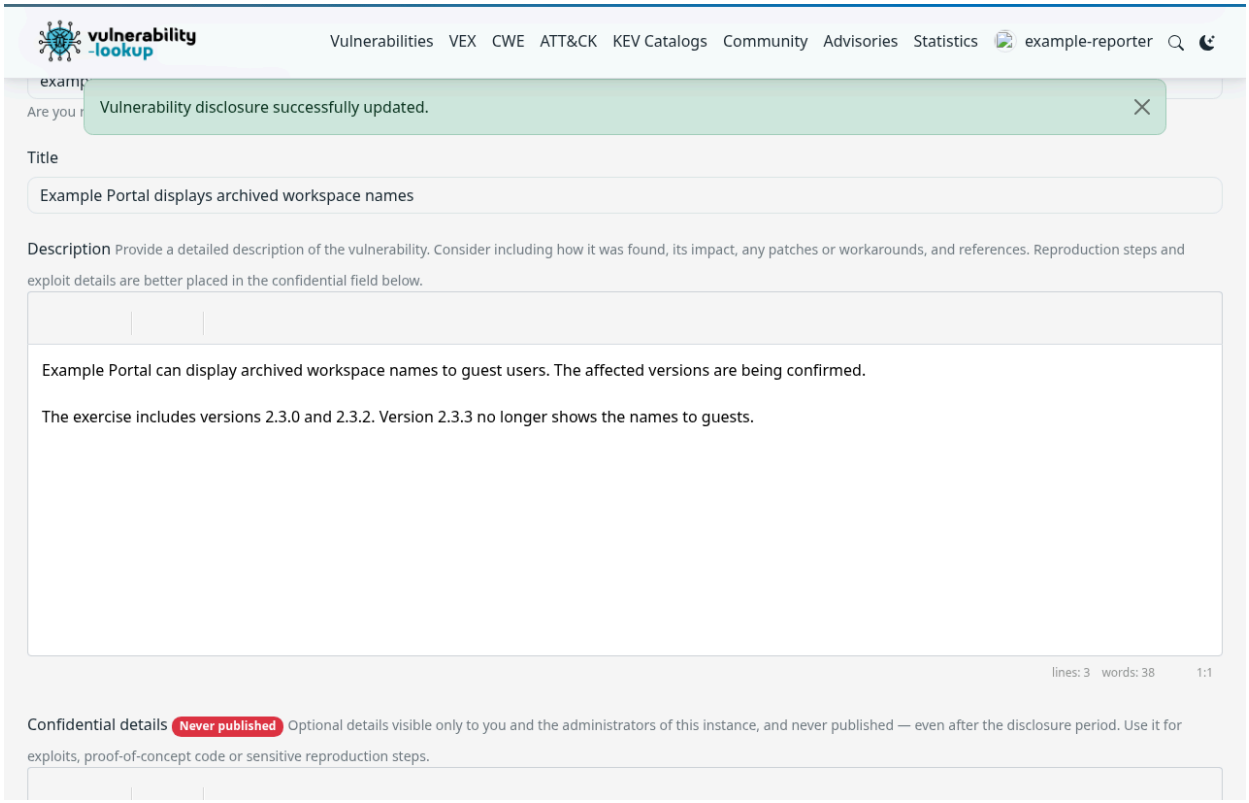


Figure 3.6: The reporter can edit the report once the case is waiting for their input.

Use the queue's **Reply needed** filter when following up. An open case is flagged when the latest message is from the reporter. State changes and report edits appear in the discussion history alongside messages. When the requested information arrives, review it and move the case back to **Under review**. Admin queue and review · Discussion

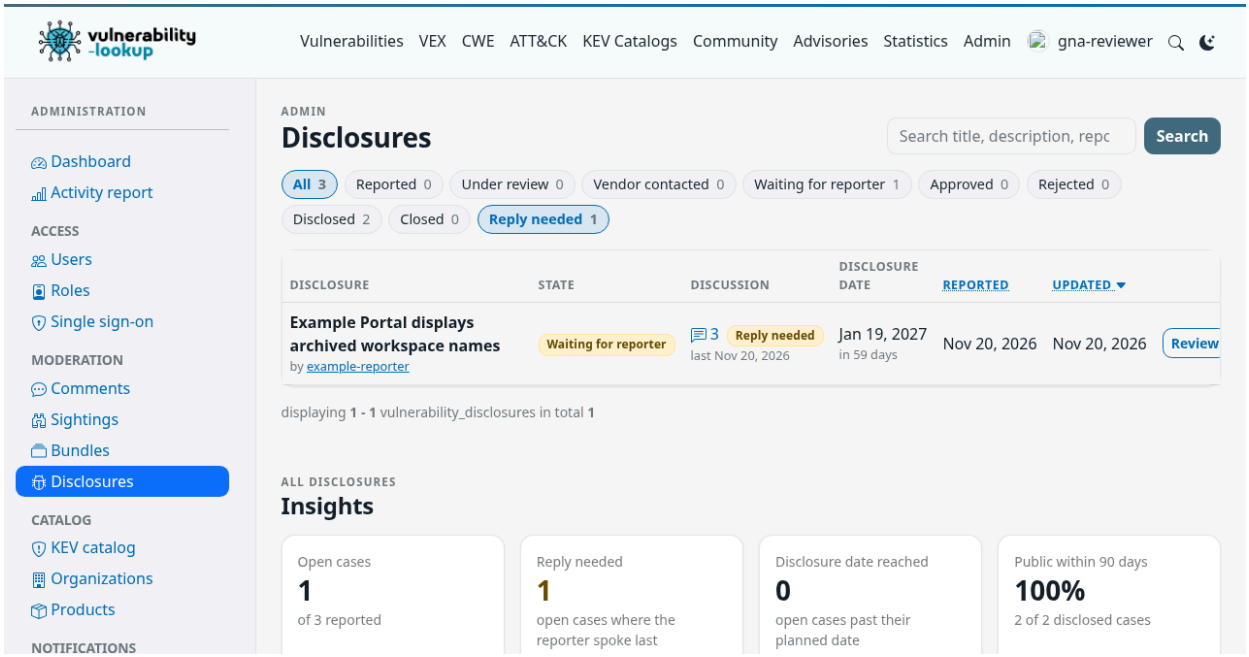


Figure 3.7: The queue brings cases with a reporter message awaiting reply to the coordinator's attention.

Done when the reporter has an acknowledgement and the case has a documented triage decision, an owner, and a next action.

BCP-02 · Receiving and Handling Vulnerability Reports

CHAPTER 04

Investigation and coordination

4.1 Agree on scope and protection

1. Ask the maintainer to confirm affected products and versions, user impact, and available mitigations. Record the vendor contact and move the case to *Vendor contacted* after making contact.
2. Track the fix or mitigation, test status, fixed versions, and update availability. In **Discussion**, use **Post message** to update the reporter. Messages support plain text and are sent by e-mail to the other side. State changes, date changes, and edits appear in the same history.

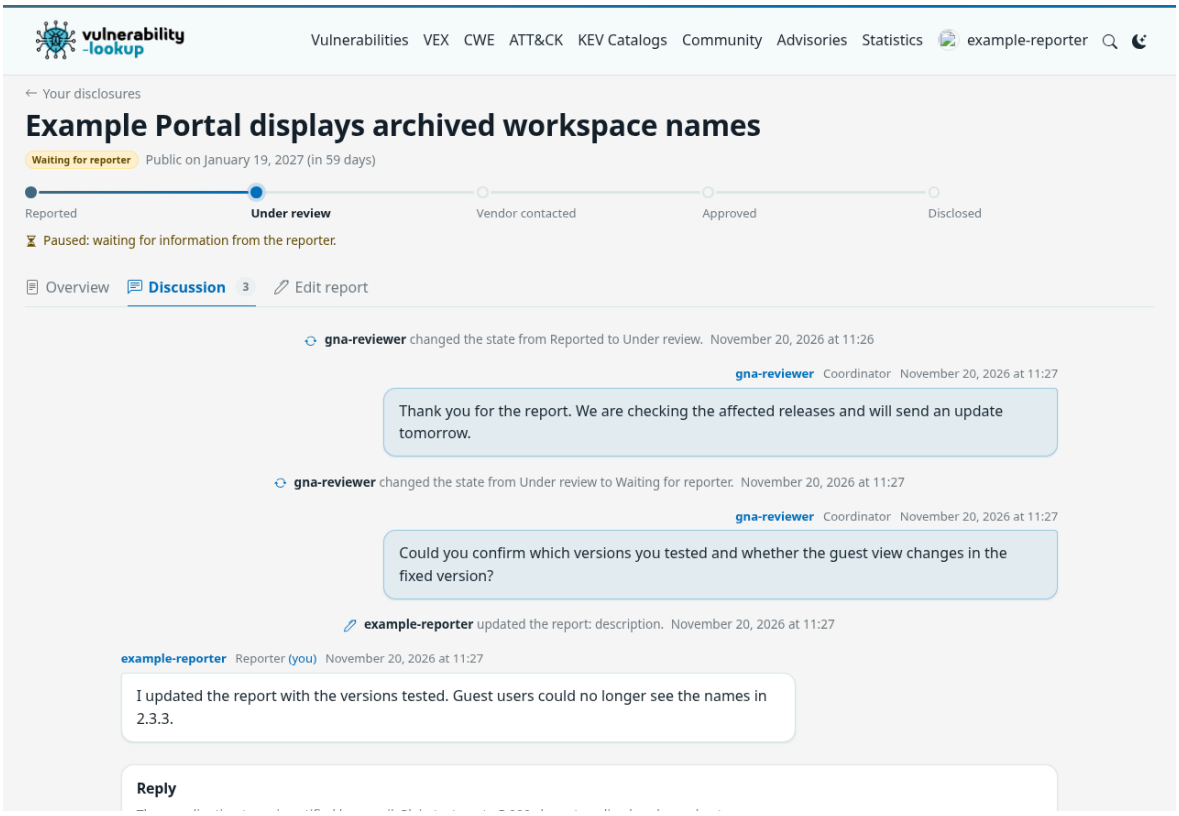


Figure 4.1: The reporter and coordination team can follow their messages and case changes in one thread.

3. Use **Notes → Add note** for vendor calls, internal findings, and the next action. Managers can read the notes and authors can delete their own notes. Notes stay outside the shared discussion and send no e-mail. If the vendor correspondence lives in another system, enter its ticket number or URL as **External reference** in **Review**. That field is visible to managers and in the admin queue.

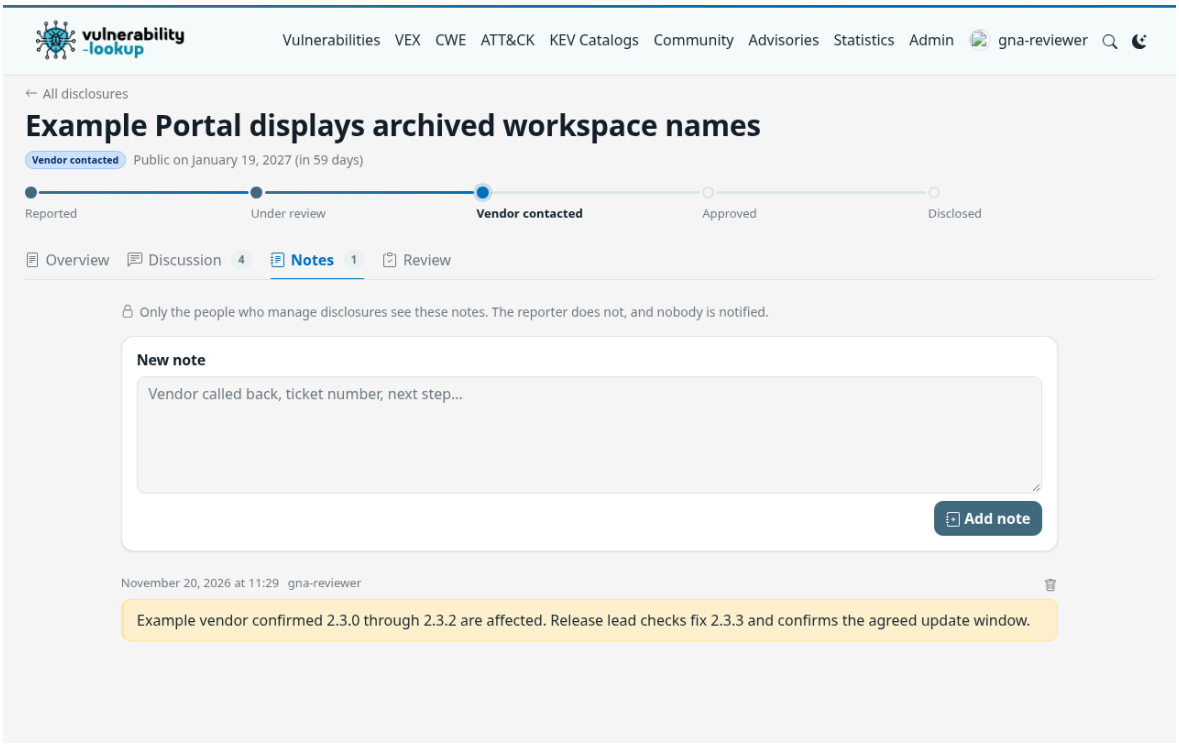


Figure 4.2: The coordination team records the confirmed scope and release handoff in a private note.

4. Agree on a publication time and time zone, the identifier owner, the advisory owner, and the next check-in. Give each participating team a contact for changes in scope or timing. Discussion and private notes · External reference

4.2 Decide when to disclose

In **Review**, set **Disclosure date** to the agreed target. The initial value is 60 days after the report was created. Revisit the plan if there is a leak, evidence of exploitation, or a delay in delivering protection. If an early warning is needed, record the reason and the protection users can apply now. Route applicable reporting obligations through your organisation's reporting process in parallel. CRA reporting

Set **State** to **Approved** and select **Save review** when the case is ready for the release handoff. The case stays private even after its target date passes. Coordinate the explicit transition to **Disclosed** with the advisory and public vulnerability record. CVD lifecycle

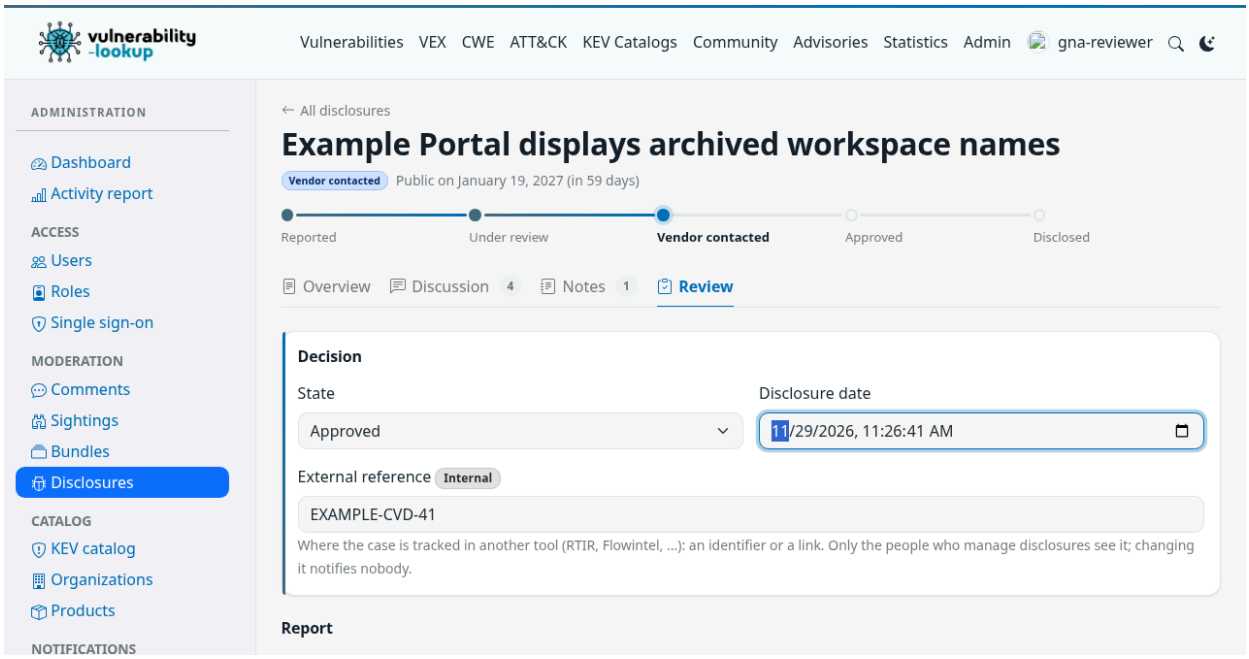


Figure 4.3: Review records the planned date, approval, and a manager-only link to the vendor's case.

Use **Insights** below **Admin → Disclosures** to spot approaching dates, dates already reached, and replies still owed. Review median time to first response and to publication to identify delays in the team's process. The timeline metrics depend on the review history recorded for each case. CVD insights

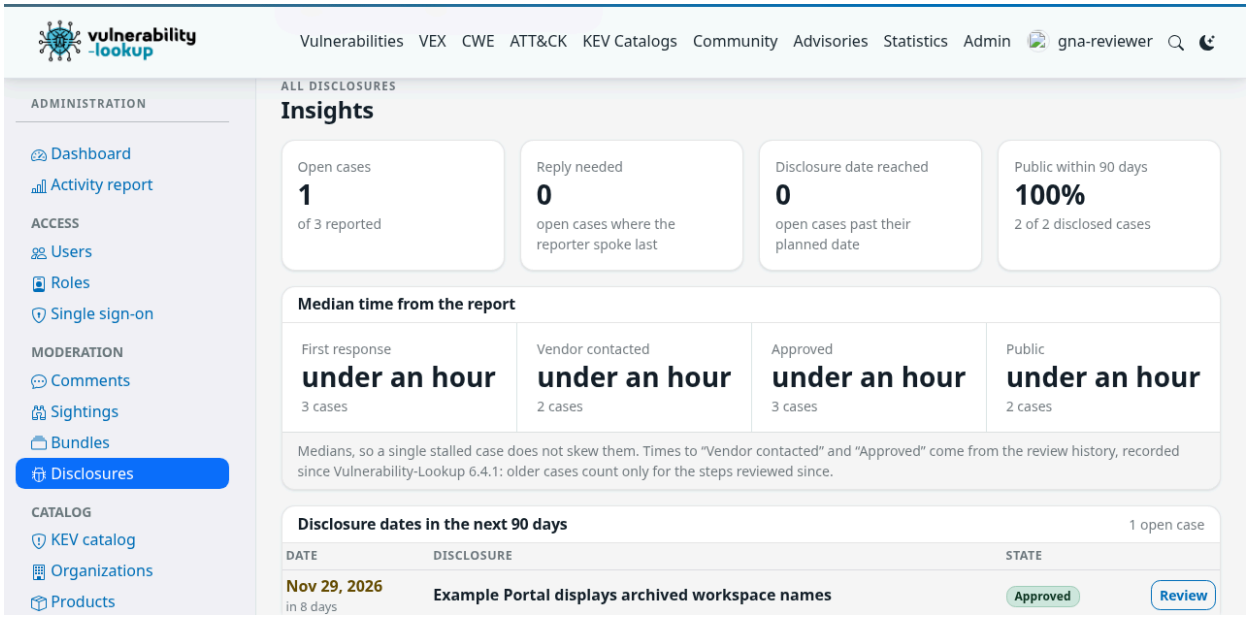


Figure 4.4: The admin-only Insights view lists upcoming release dates and queue indicators.

Done when the release approver has a public draft, confirmed remediation information or an agreed early warning, and a named owner for every release step.

BCP-02 · Investigating and Resolving Vulnerabilities, Communicating with Submitters and Users, and Coordinated Vulnerability Disclosure

PART 03

Publishing and Follow-up

CNA and GNA publication, advisories and corrections.

CHAPTER 05

Write and evaluate a vulnerability record

A good vulnerability record tells readers whether they are affected, what the impact is, and how to protect themselves. Use Vulniverse to turn verified findings into structured fields. Check that the record, the public advisory, and the fix documentation agree before the release.

The walkthrough uses a fictional Example Workspace draft in a local Vulnerability-Lookup 7.1.0 development instance. Its versions, scores, credit names, and URLs illustrate the editor. The separate CPE lookup uses a real catalogue entry to show how to verify a product identifier. Vulnerability-Lookup revision b4c82c1

5.1 Prepare the title, description, and affected scope

1. Sign in as a publication operator and open **Advisories → New vulnerability advisory**. Use the CVE or GCVE profile for your publication route. To correct an existing record, open it with **Edit** and retain its identifier.
2. In **Summary**, write a title that names the product or component and the issue. In the description, state the affected behaviour, the relevant conditions, who or what can be affected, and the resulting impact. Use precise version and configuration language. Put private evidence and coordination details in the restricted case.
3. In **Classification**, add the most specific CWE supported by the root cause. For the example, the missing authorisation check justifies CWE-862. Document user impact in the description and add an **Impacts** entry when it adds detail. Add a CAPEC mapping only when the evidence supports the pattern. CWE-862 · CVE Record Format

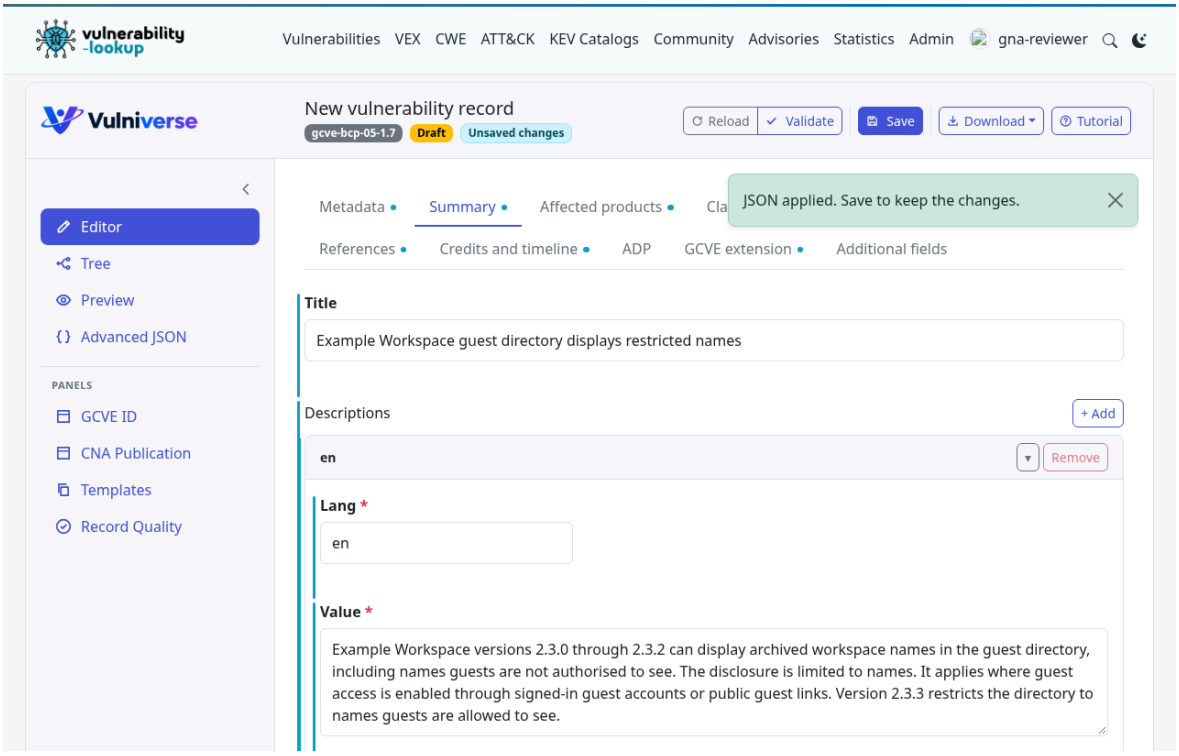


Figure 5.1: The summary describes the affected feature, conditions, impact, and fixed release.

4. In **Affected products**, use the vendor and product names readers will recognise. Give each independently versioned product or branch its own entry or range. Use the product's version scheme. In the example, `version: 2.3.0` and `lessThan: 2.3.3` mark an affected range with an inclusive start and an exclusive end. Version 2.3.3 is explicitly confirmed unaffected. Leave unverified releases at `unknown` when setting the default status. Check separately for backports and distribution-specific version numbers. CVE Record Format

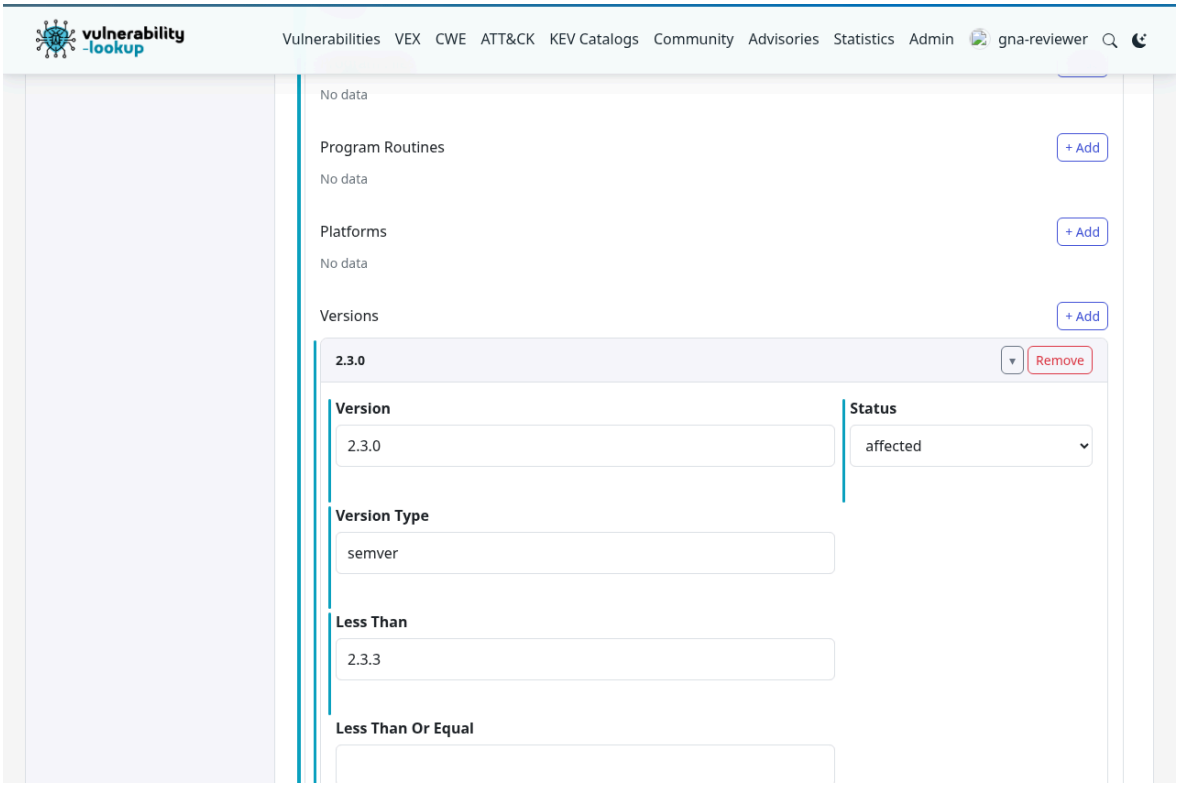


Figure 5.2: Use structured version ranges so consumers can match their installed release.

5.1.1 Look up an existing CPE

1. On **cpe.gcve.eu**, search by vendor and product, and narrow by **Part** for applications, operating systems, or hardware. You can also search by PURL when you know the package coordinates. Open the matching entry and check the exact vendor, product, version, edition, target platform, and deprecation status. If the entry is deprecated, verify that its replacement identifies the affected product. CPE catalogue search
2. In **Affected products**, add the canonical CPE 2.3 value to the matching product's **CPEs**. Record the affected versions separately and keep them consistent with the advisory. For platform-dependent combinations, use **CPE Applicability** if you can verify the conditions that must hold together. Product and vendor suggestions in Vulniverse depend on the instance's CPE Editor settings. Vulniverse configuration
3. If there is no exact CPE, keep the verified vendor, product, and version data. Add the verified package identity where the schema supports it and propose a catalogue entry if appropriate. The fictional Example Workspace record leaves CPE blank. The screenshot uses a separate catalogue entry. Mozilla Firefox CPE entry

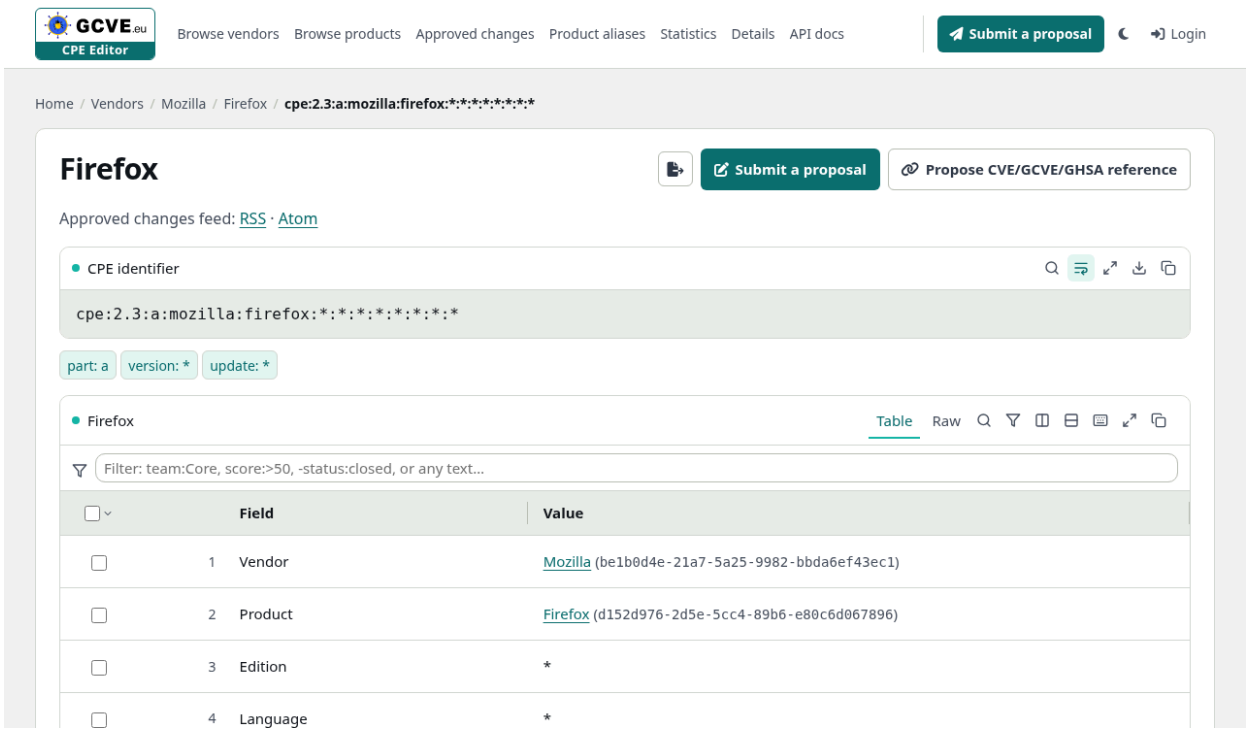


Figure 5.3: The catalogue entry shows the exact CPE and its platform attributes. This Firefox entry illustrates lookup only.

5.2 Record severity and response priority

5.2.1 Add a CVSS assessment for each supported scenario

Open **Severity**, select **Add metric**, and choose the CVSS version required by your publication policy. In CVSS 4.0, use **Calculate** or enter the complete vector and corresponding base score and severity. The vector records your assumptions about factors such as access, privileges, and the affected systems. When publishing CVSS 4.0 scores, label the metric groups used. Use CVSS-B for base, CVSS-BT when including threat metrics, CVSS-BE when including environmental metrics, and CVSS-BTE when including both. Explain the context used for threat and environmental assessments. FIRST CVSS 4.0 specification

When assessors disagree, compare the versions, assumptions, and environments behind the vectors. If both assessments describe verified scenarios, create separate metric entries and name each scenario. On the editor revision shown, use **Advanced JSON** to set each metric's `scenarios` list with a language and a descriptive value, then **Apply JSON**. Use **GENERAL** for a baseline intended to apply when no more specific scenario matches. Keep a partner's independent assessment attributed to its source, such as in that partner's own ADP container when it publishes one. For conflicting assessments of the same scenario, record the rationale and agree which assessment the publisher will use. CVE Record Format

In the fictional example, the publisher uses two CVSS 4.0 base metrics. Signed-in guest accounts require low privileges (PR:L, 5.3). An optional public guest link applies to users without an account (PR:N, 6.9). Publish both scenarios with their vectors and scores so users can tell which applies to their deployment.

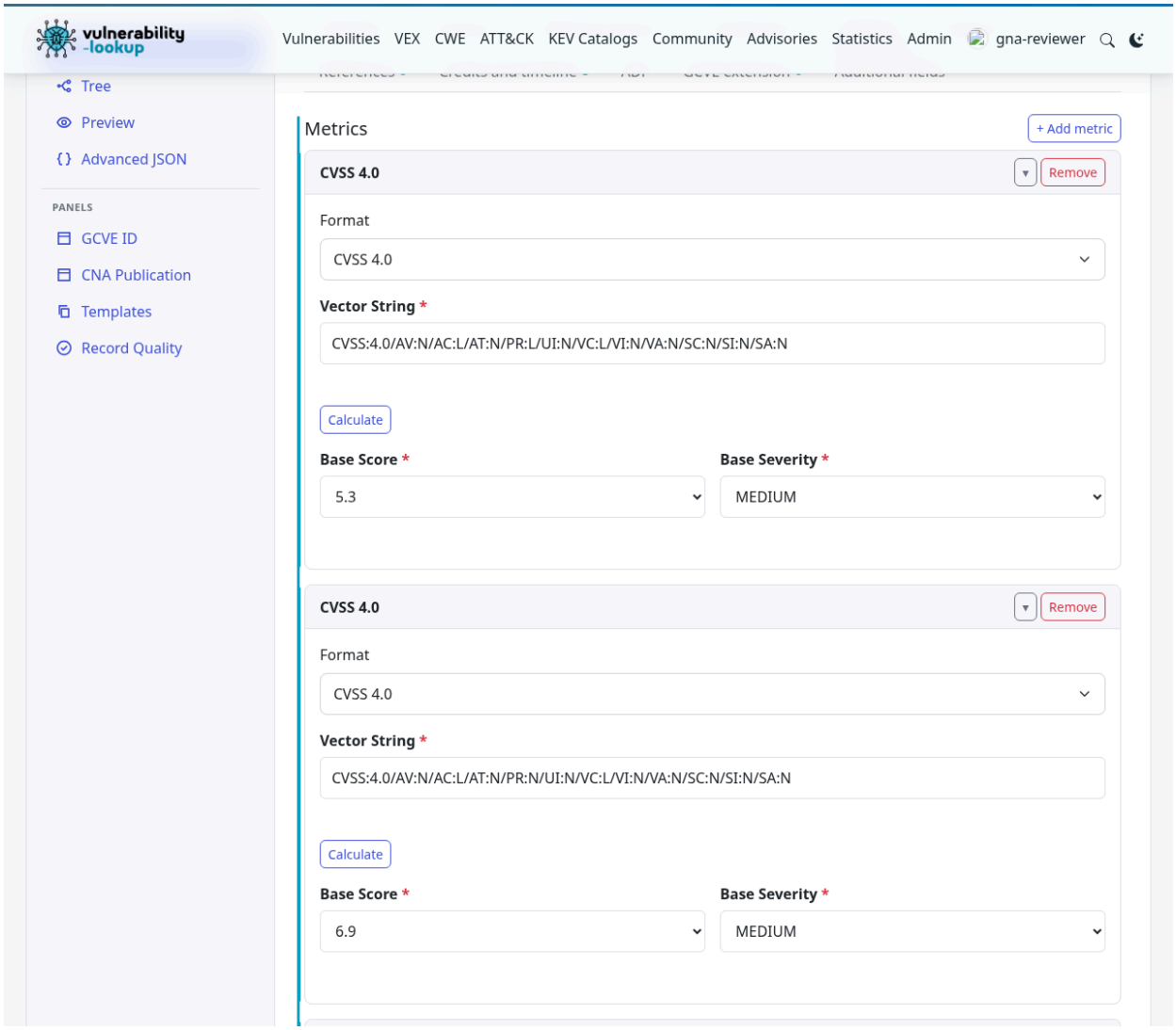


Figure 5.4: The Severity tab shows two CVSS 4.0 metrics for the same record.

5.2.2 Add an SSVC decision

Use SSVC to record response priority for a named stakeholder role. Choose the decision model that matches the party making the decision. For example, the Supplier model considers exploitation, automatable exploitation, and technical impact. Record the assessment time, the evidence behind each decision point, and the resulting action in your coordination record or advisory as appropriate. The fictional Supplier options below lead to **Scheduled**. Revisit the decision when exploitation evidence or other inputs change. CERT/CC SSVC Supplier decision model · Track exploitation in a local KEV catalogue

For a CVE or GCVE record in this editor revision, use **Advanced JSON** to add a metric with `format` set to SSVC and an `other` object whose `type` is SSVC and whose `content` is the structured assessment. Add it alongside the existing metrics under `containers.cna.metrics`. Preserve the object structure when editing it. The example below uses the SSVC interchange schema version and the Supplier decision points. Use values supported by your own evidence. CERT/CC SSVC schema · Patch2Vuln's structured output

```
{
  "format": "SSVC",
  "other": {
    "type": "SSVC",
    "content": {
      "version": "2.0.3",
      "role": "Supplier",
      "timestamp": "2026-11-20T12:00:00Z",
      "options": [
```

```
          {"Exploitation": "none"},
          {"Automatable": "yes"},
          {"Technical Impact": "partial"}
        ]
      }
    }
  }
```

5.3 Give users a fix and credit the contributors

In **Guidance**, state the first fixed release for each supported branch and how users should obtain or apply it. Document temporary workarounds and their limits. In **References**, use **Add reference** and tag each approved public link for the job it performs.

Tag	Use it for
`patch`	A public fix commit, patch, or backport that addresses this vulnerability. Verify that it fixes the affected branch and include an approved backport link when users need it.
`vendor-advisory`	The vendor's advisory for this vulnerability.
`release-notes`	The release documentation that identifies the fixed version.
`mitigation`	Specific instructions that reduce exposure while users prepare to patch.
`third-party-advisory`	A separate advisory from a coordinator, distributor, or other third party.

Link the patch alongside the advisory and release notes when it is public. The patch helps downstream maintainers identify the precise fix, check whether it was backported, and avoid confusing it with a different change in the same release. If the fix is available only through a vendor channel, link the approved advisory or release notes and state how eligible users obtain the fix. Verify each URL while signed out. Keep private tracker and repository links in the restricted case. CVE Record Format

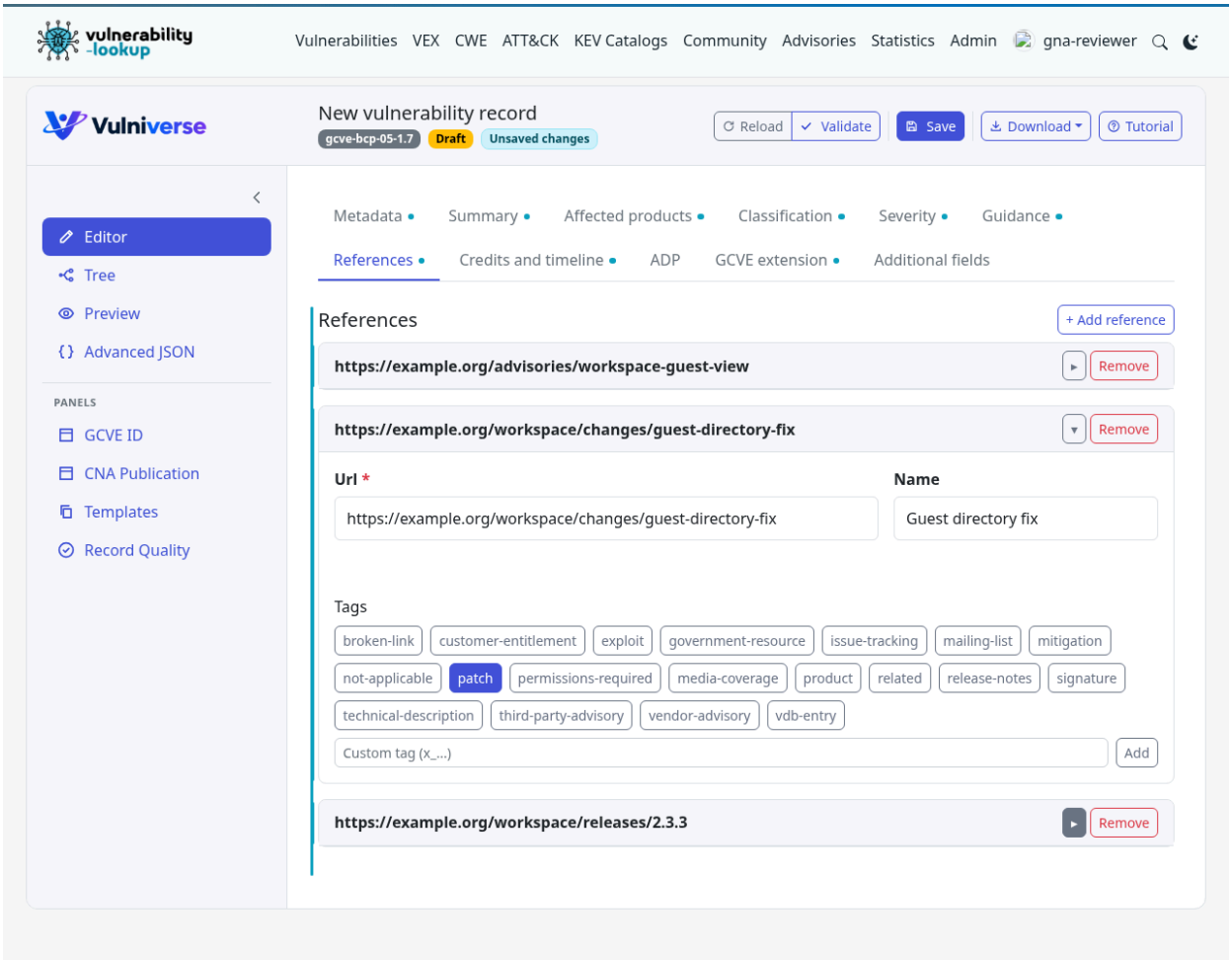


Figure 5.5: The record links the vendor advisory, exact patch, and release notes.

In **Credits and timeline**, confirm each contributor's preferred public name and role before adding their credit. Distinguish the person who found the issue, the reporter who brought it to the team, and the remediation developer or coordinator where applicable. Respect requests for anonymity and copy only wording approved for publication. Record relevant dates with their meaning, such as discovery, fix availability, and public release. Check credits and dates against the advisory. GCVE BCP-02 recognition guidance

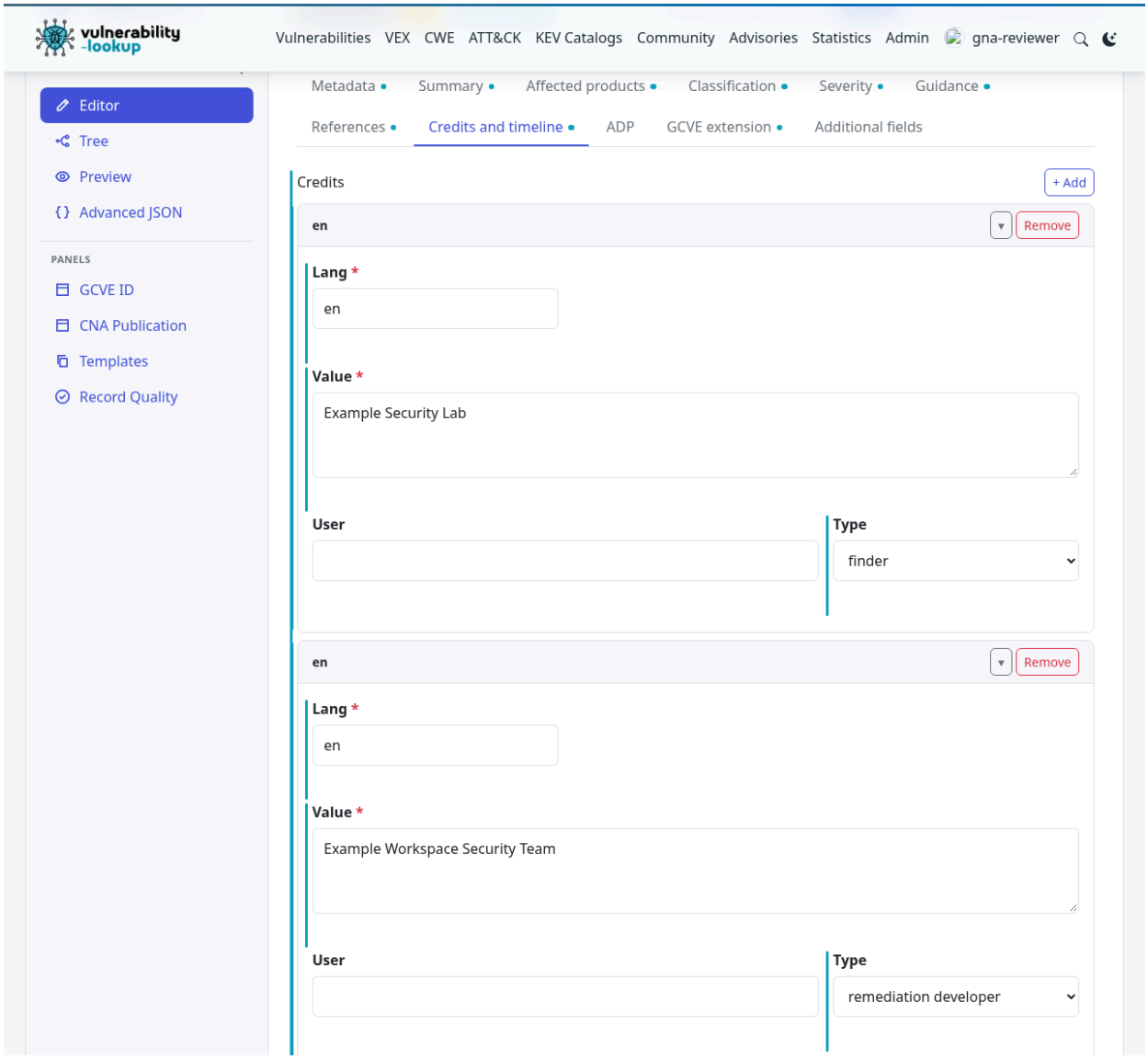


Figure 5.6: The publisher records approved credits with their actual roles.

5.4 Use Record Quality to find missing fields

Open **Record Quality** in Vulniverse's sidebar. The panel evaluates the draft in the editor using the Vulnerability-Lookup assigner scorecard rules for CVE and GCVE records. If the panel is hidden, check `panels.record-quality` in `config/vulniverse.json`. Each pillar contributes up to 25 points. Open the panel after edits or click **Recalculate** to see which checks receive full, partial, or zero credit. Assigner scorecard rules · Record Quality panel

Pillar	What the checks look for
Describe	A meaningful English description, a title, and a CWE.
Scope	Affected vendor and product, version information, and a CPE or applicability block.
Assess	A severity statement, a modern assessment such as CVSS 4.0 or SSVC, and structured impact.
Act	References, reference tags, a remediation reference, and a solution or workaround. The `patch` tag earns full remediation credit. The `vendor-advisory`, `mitigation`, and `release-notes` tags earn partial remediation credit if a patch link is unavailable.

The initial Example Workspace draft scores **75/100**. An untagged link earns reference credit and leaves the tag, remediation, and solutions checks empty. After the publisher adds the patch reference, tags the links, and fills in the fix and workaround, **Act** rises from **7/25** to **25/25** and the record reaches **93/100**. The fictional product has no verified CPE, so the **Scope** score stays at **18/25**.

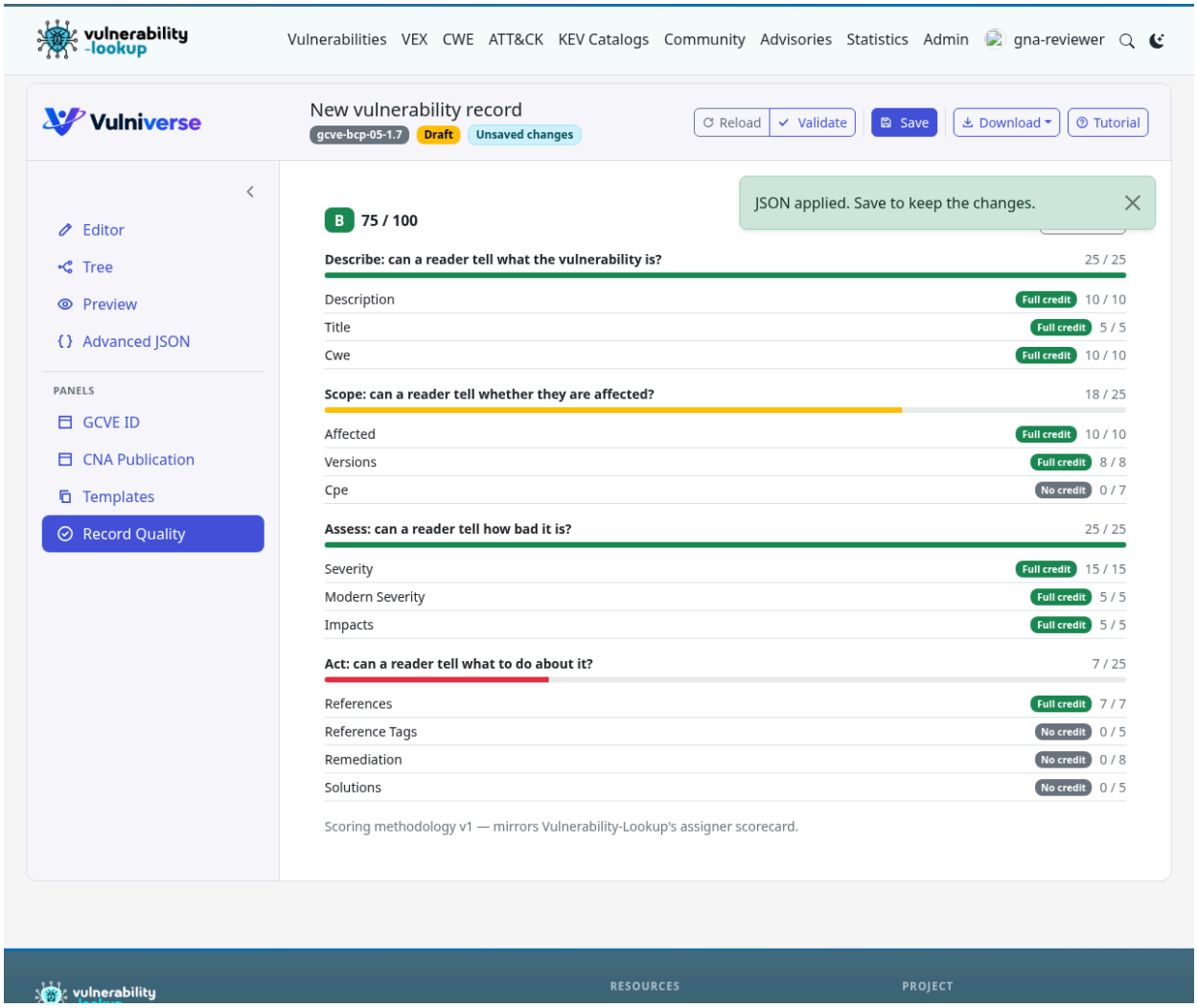


Figure 5.7: The initial evaluation identifies the missing tags and fix guidance.

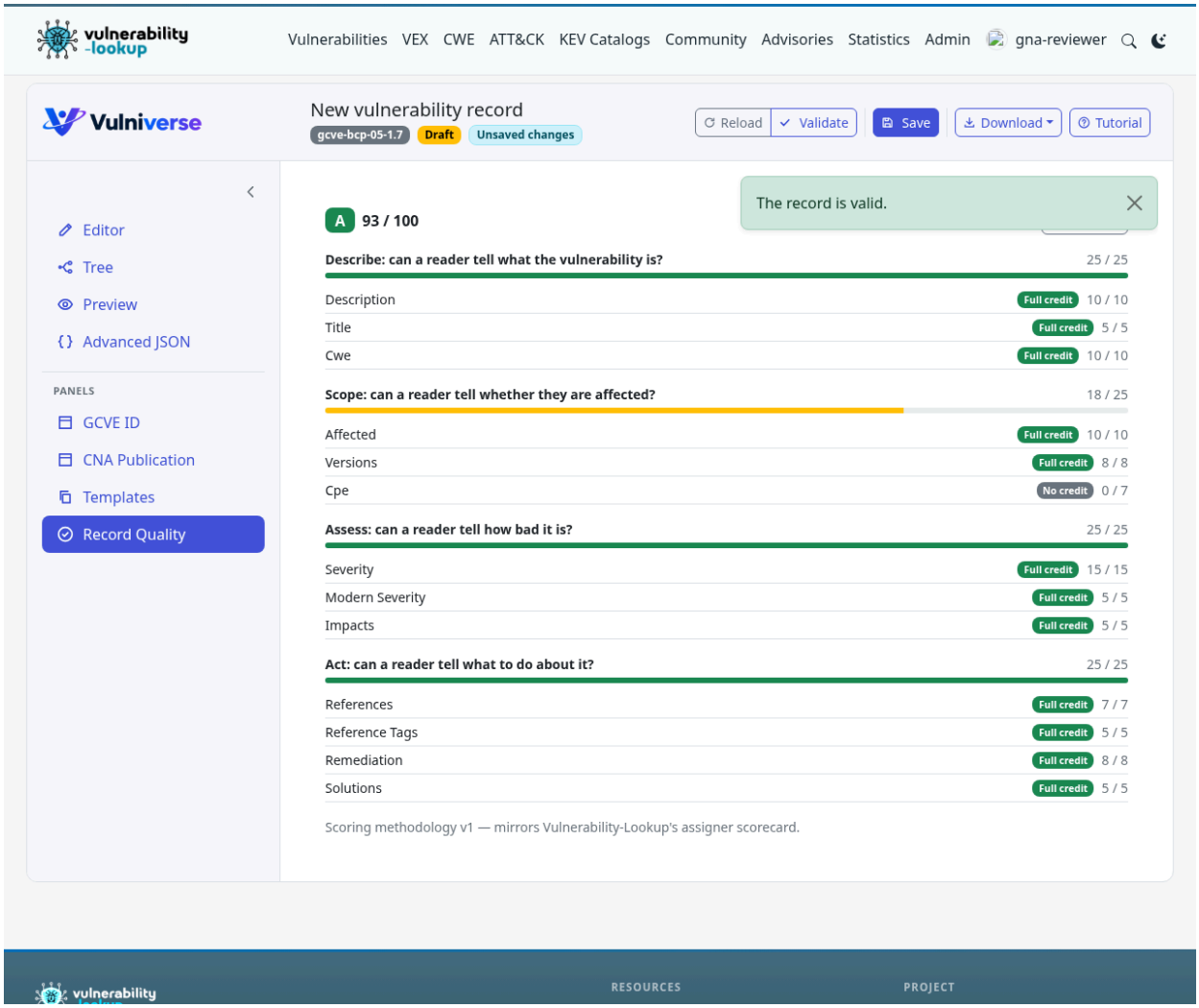


Figure 5.8: Recalculating after the corrections shows the effect on the same record.

Use the score to find incomplete fields. Verify factual accuracy, reference destinations, applicability, and release approval independently. Credits and timeline details still need a human check. Check **Preview** for what readers will see and use **Validate** to check the format. Keep embargoed drafts in the restricted workflow until release, since **Save** can publish a new local record. After release, check the signed-out record and advisory for the same versions, links, and instructions. CNA publication · GNA publication · Advisory and follow-up

CHAPTER 06

Publish as a CNA

A CVE Numbering Authority (CNA) assigns CVE IDs within its authorised scope and publishes CVE records through CVE Services. Vulnerability-Lookup keeps the private disclosure case, the local vulnerability record, and the CNA publication workflow in one place. The disclosure case controls who can see the report. The local record supplies the CNA container. The publication workflow tracks reserve, publish, update, and reject requests to CVE Services. GCVE BCP-02 · CNA Publication Service

The screenshots show the Vulniverse editor in Vulnerability-Lookup 7.1.0 development revision `b4c82c1`. The walkthrough uses fictional data, a local CVE Services simulator, and GNA 65535, the GCVE test namespace. For an authorised service rehearsal, use the CVE Services test environment and credentials issued for it.

6.1 Choose a publication path

Organisations can publish advisories in Vulnerability-Lookup as soon as their instance and advisories are ready. A local identifier such as `EXAMPLE-SA-2026-0001` keeps the advisory addressable while the organisation decides whether it needs authority to assign CVE or GCVE IDs.

Goal	Path
Publish advisories under your own identifiers	Start with local publication.
Assign CVE IDs and publish records through CVE Services within an agreed scope	Apply to become a CNA.
Assign GCVE IDs in your own namespace and publish a discoverable feed	Apply to become a GNA.

We recommend that eligible organisations, including software, hardware and service vendors, apply for both CNA and GNA status. The roles let a team publish through the CVE Program and operate its own GCVE publication source. Eligibility and approval are assessed separately. Choose the identifiers and channels each case needs, and manage both routes from the same Vulnerability-Lookup instance when appropriate. Coordinate CNA and GNA publication on one instance

6.1.1 Start with local publication

1. Enable user accounts and configure `local_instance_name`, a stable `local_instance_uuid`, and `local_instance_vulnid_pattern` in `config/generic.json`. For IDs such as `EXAMPLE-SA-2026-0001`, set `local_instance_vulnid_example` to `EXAMPLE-SA-yyyy-nnnn` and use a matching pattern. Leave `cna` disabled. Minimal publication instance
2. With an authorised account, use your instance's local API to create the year's range if needed, reserve a local ID, and publish the record at release time. Use `POST /api/cna/cve-id-range/<year>`, `POST /api/cna/cve-id`, and `POST /api/cna/cve/<id>`, respectively. These routes use local authentication and your configured identifier format. See the instance's `/api/` reference for parameters and required permissions. Verify the advisory while signed out. Vulnerability-Lookup API
3. Keep the local identifier and advisory URL stable. If you later assign or receive a CVE or GCVE identifier for the issue, add it to the advisory and link the published records. You can also publish locally and work with an existing CNA or GNA when a case calls for an official identifier.

6.2 Become a CNA

1. **Prepare your policy and proposed scope.** The CVE Program asks for a public vulnerability disclosure policy, a public source for new disclosures, and agreement to its Terms of Use. Prepare the policy and advisory URLs, an organisational contact, the products or services you propose to cover, and an estimate of annual CVE ID needs. CVE Program partner requirements · CVE Program Terms of Use
2. **Submit the registration form.** In the CVE Program Web Forms portal, choose **Register to Become a CNA**. Use the contact and proposed scope from your preparation and keep the ticket number for follow-up. How to Become a Partner · CVE Program Web Forms
3. **Complete onboarding.** Confirm your agreed scope during onboarding, attend the introductory session, and complete the practice CVE exercises. Once the CVE Program lists your organisation as a partner, confirm your registered short name and obtain the appropriate CVE Services access before configuring production publication. CVE Program onboarding · List of Partners

The CVE Program's 2019 article describes the original registration and training process. Use the current partner page and web forms portal for the application. Become a CNA to Assign Your Own CVE IDs

6.3 The workflow

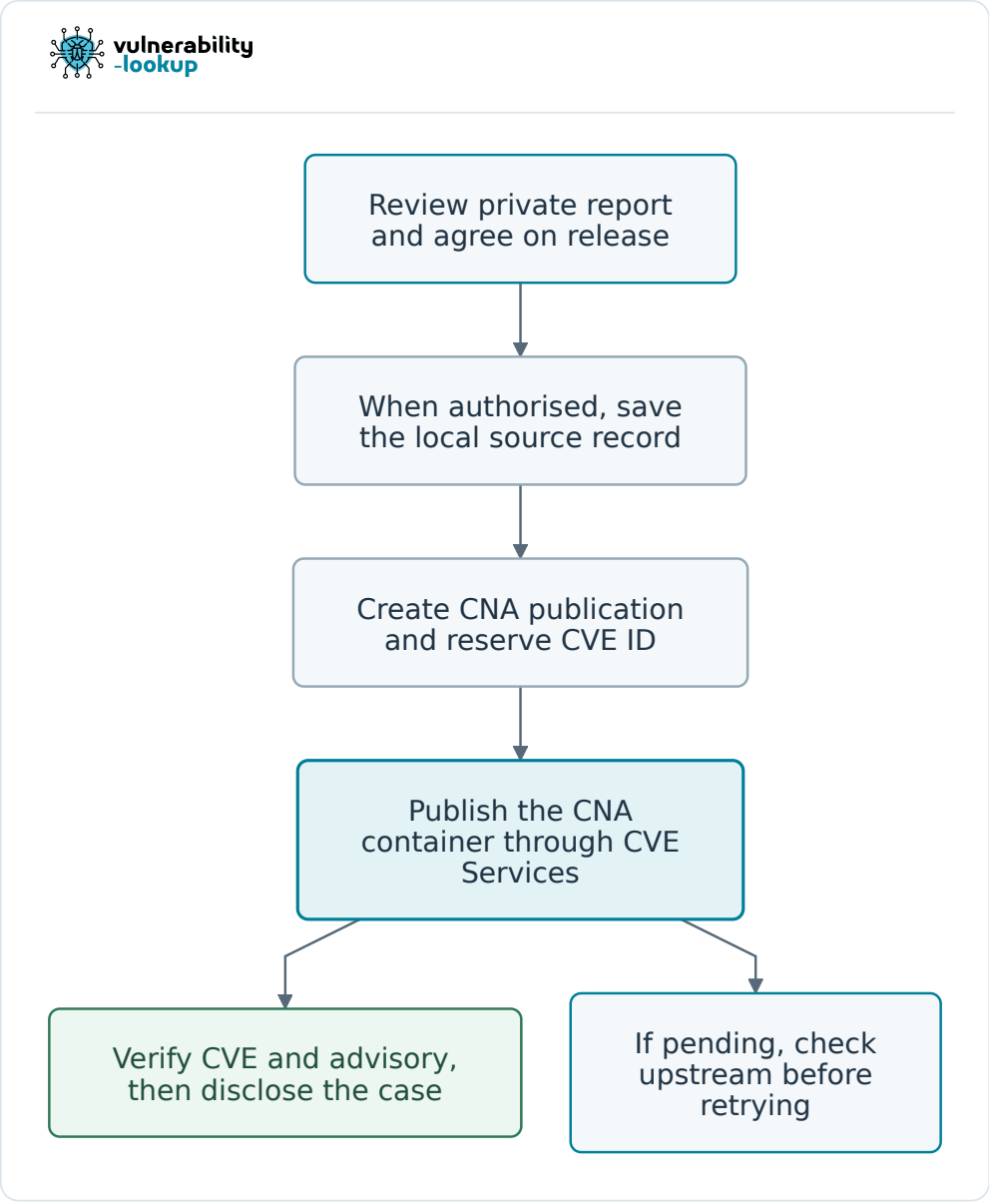


Figure 6.1: The CNA workflow, from private intake to CVE publication and verification.

6.4 Set up the CNA

1. Confirm your authority, scope, and registered CNA short name. Agree who can approve releases, who can submit them, and who handles corrections. For a publication-only deployment, follow the installation guide for PostgreSQL, Valkey, Kvrocks, and the web service. Minimal CNA/GNA Publication Instance
2. In `config/generic.json`, enable `user_accounts` and set `cna` to `true`. Configure the instance's local source and stable UUID. If the organisation also runs a GNA, configure the local source for its assigned GNA number. Set up the GNA
3. Copy `config/cna.json.sample` to `config/cna.json`. Set `CVE_URL` to the authorised environment's API base URL and `SHORT_NAME` to the CNA's short name. Use `https://cveawg-test.mitre.org/api` for the test service and `https://cveawg.mitre.org/api` for production. Generate a Fernet key for `CNA_CREDENTIAL_ENCRYPTION_KEY` and store it securely. Keep the key available for recovery of the encrypted per-user credentials. CNA configuration
4. In `config/website.py`, enable `vulnerability_disclosure` and `vulnerability_editor` in `WEB_MODULES`. Set `VULNERABILITY_EDITOR = "vulniverse"` for this walkthrough and keep the `cna-publication` panel enabled in `config/vulniverse.json`. Apply migrations, restart the website, and verify that the disclosure inbox and configured mail delivery work. Disclosure configuration · Vulnerability editor

6.4.1 Set up operator access

Each publishing operator opens their account menu and selects **CNA credentials**, then enters their own CVE API organisation, user, and key. On the edit form, leaving **CVE API Key** blank keeps the stored key. Administrators can disable a credential at `/admin/cna-credentials` when an operator leaves or needs their publication access suspended. Per-user credentials

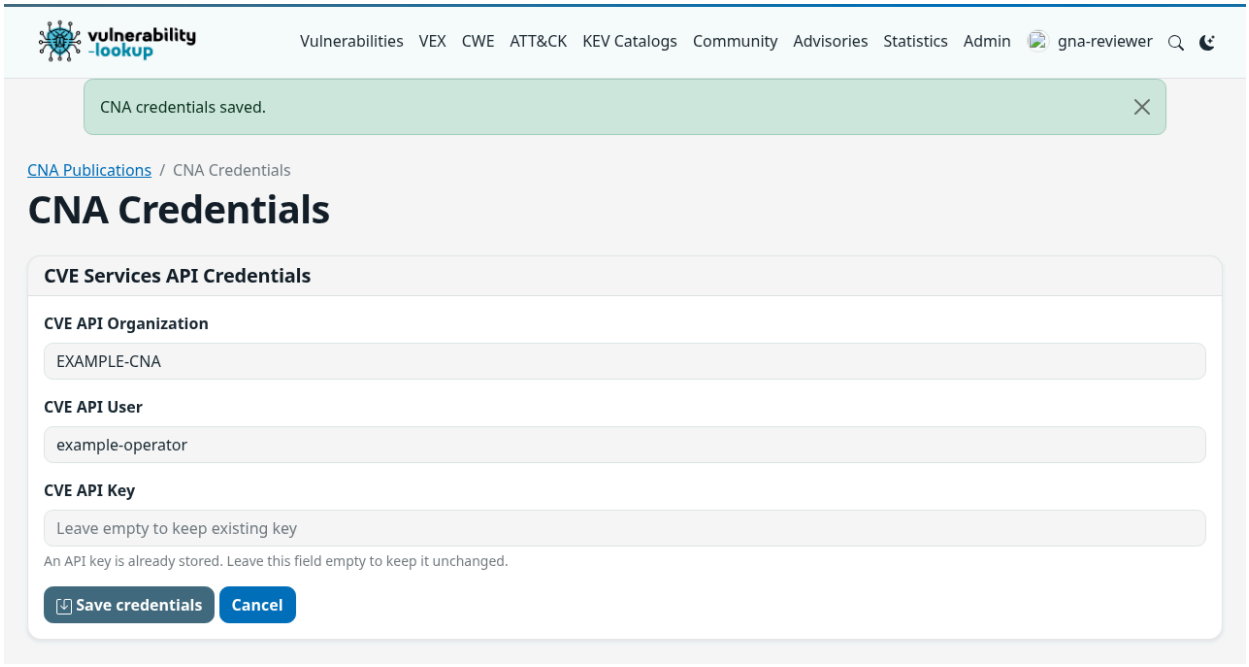


Figure 6.2: The operator saves their own CNA credentials.

Task	Access to check
Report and coordinate disclosures	`vulnerability_disclosure:report` for submission and `vulnerability_disclosure:review` for coordination. Use an administrator account for the admin queue and Review form.
Create or edit the local source record	`vulnerability:create` and `vulnerability:modify` on the relevant record.

Task	Access to check
Open or create a CNA publication	`cna_publication:view` and `cna_publication:create`, as appropriate.
Submit to CVE Services	The publication UI exposes upstream actions to administrators in this release. Check `cna_publication:reserve`, `cna_publication:publish`, and `cna_publication:reject` for the applicable action and verify that the operator has enabled credentials.

Test the reporting role separately. The built-in `reporter` role also grants local vulnerability record permissions in this release. Use a dedicated role with the report permission for external submitters. Roles and permissions

6.5 Receive and coordinate a report

1. Have the reporter sign in and open `/disclosure/new/`. Collect the affected versions, impact, and credit preference. Keep sensitive details in **Confidential details** and check that the main **Description** is suitable for eventual disclosure.
2. In **Admin → Disclosures**, assign a case owner, check scope, and search for existing CVE and GCVE identifiers. Acknowledge the report in **Discussion** and use **Notes** and **External reference** for internal coordination. If a CVE already exists, coordinate with its owning CNA before requesting another ID.
3. Confirm affected and fixed versions, mitigation, release time and time zone, and the publication owner. Record the decision in **Review** and set **State** to **Approved** when ready for the handoff. The case stays private until an operator explicitly sets it to **Disclosed**. Investigation and coordination

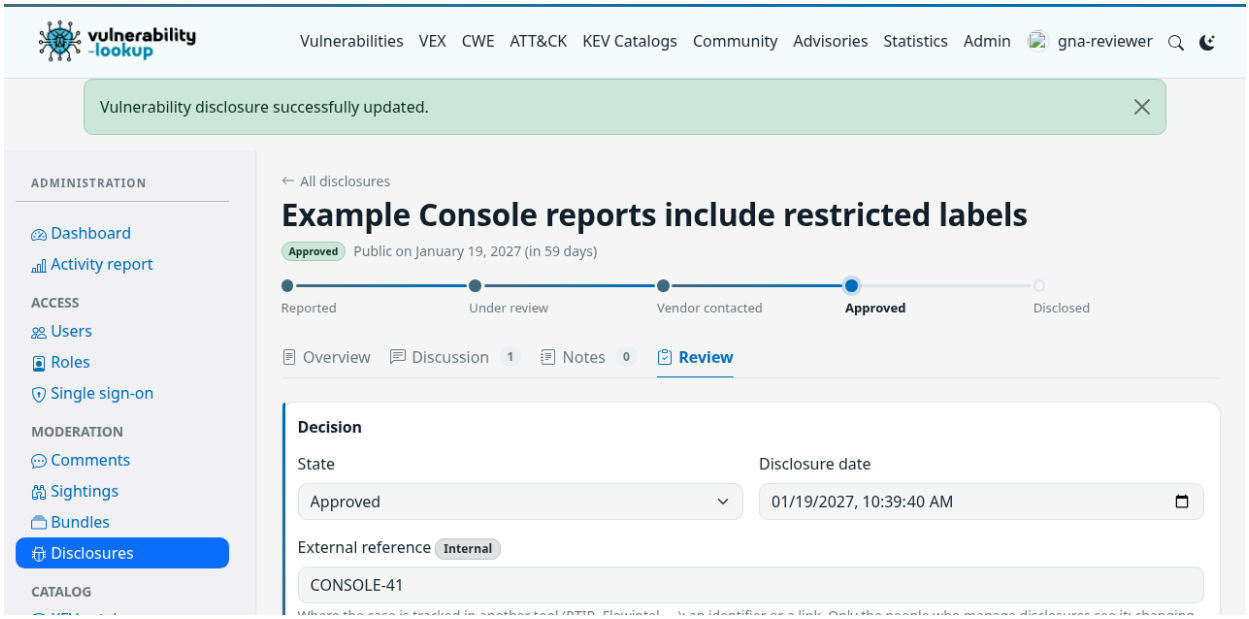


Figure 6.3: The coordinator approves the case and records its internal tracking reference.

6.6 Prepare the source record and reserve a CVE ID

1. Prepare the approved public material in a restricted workspace. Under **Advisories → New vulnerability advisory**, prepare the local source record in Vulniverse. Enter the public summary, affected versions, mitigation or fix, advisory links, and agreed credit. Review **Preview** and click **Validate**. Verify the CNA provider metadata and the container required by CVE Services. Write and evaluate a vulnerability record · CVE Record Format
2. Assign the record its local identifier. The example uses a GCVE ID allocated by the same organisation through the GNA workflow. In this Vulniverse revision, the **CNA Publication** panel uses the GCVE

identifier to find the saved source record. For a deployment using Vulnogram and a separate local identifier pattern, follow the Vulnogram publication steps.

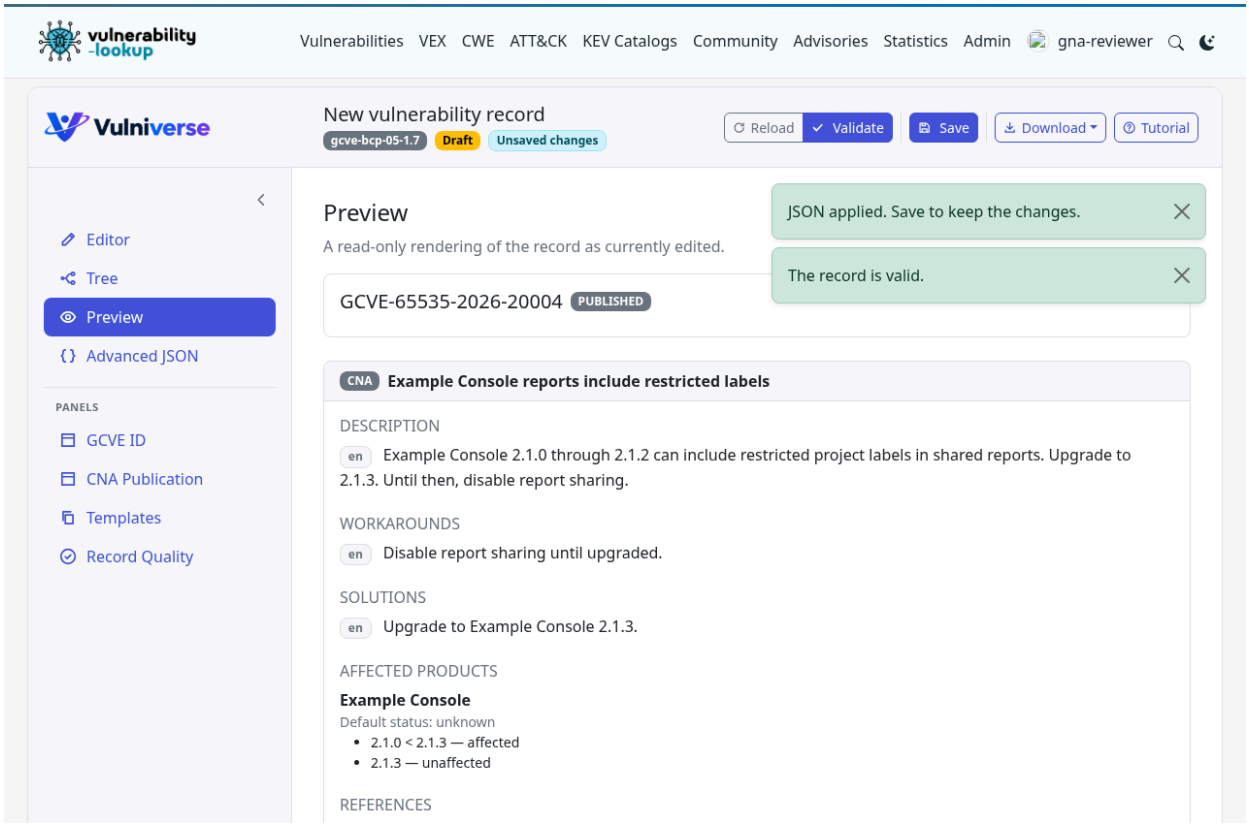


Figure 6.4: Vulniverse previews the public record and confirms local validation.

Save only when local exposure is authorised
The **CNA Publication** panel requires a saved local record. On a public Vulnerability-Lookup instance, **Save** exposes that record through the local source even while the CVD case is private and the CVE is unpublished. If the case is embargoed, keep the content in the restricted workspace until the release window. When an ID must be obtained earlier, use an authorised reservation process or an access-controlled publication instance, and plan how that reservation will be tracked without obtaining a second ID.

3. Once the local record can be saved, open **CNA Publication → Create publication** and follow **Open full publication page**. You can return to the same publication through **Advisories → CNA Publications**. Its initial state is `LOCAL_ONLY`.
4. On the publication detail page, choose **Reserve CVE ID**. Confirm `RESERVED` and write the returned CVE ID in the case and release plan. The service attempts to mirror the ID into the local record's `cveMetadata.cveId`. If the publication is `RESERVED` but reports a local sync error, reconcile the local record against the existing upstream reservation.

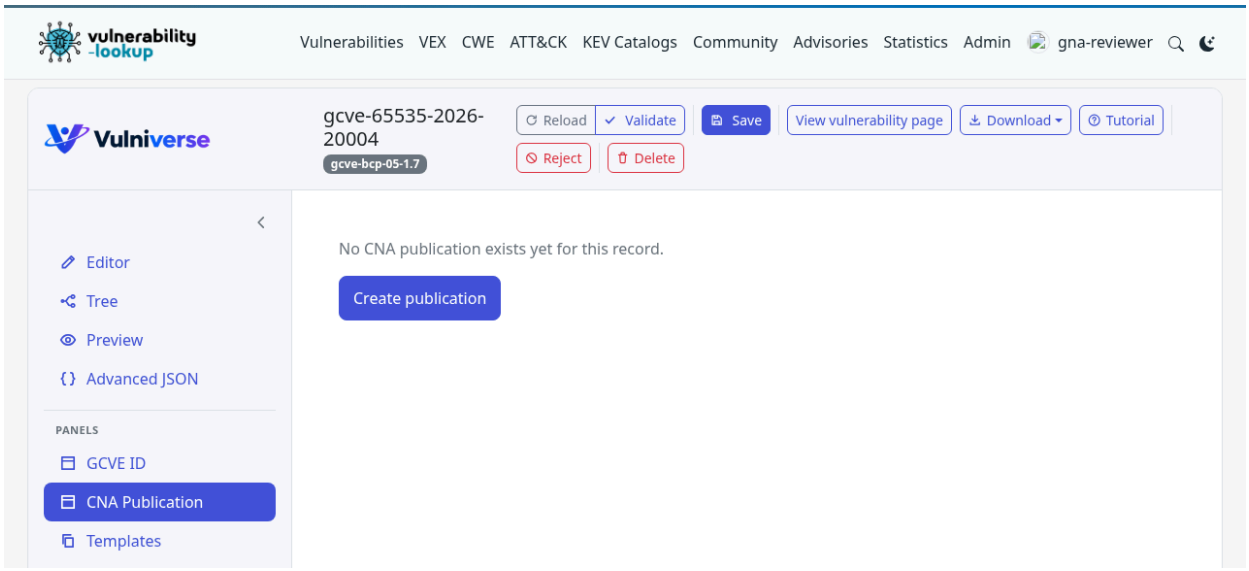


Figure 6.5: Create the CNA publication from the saved local record.

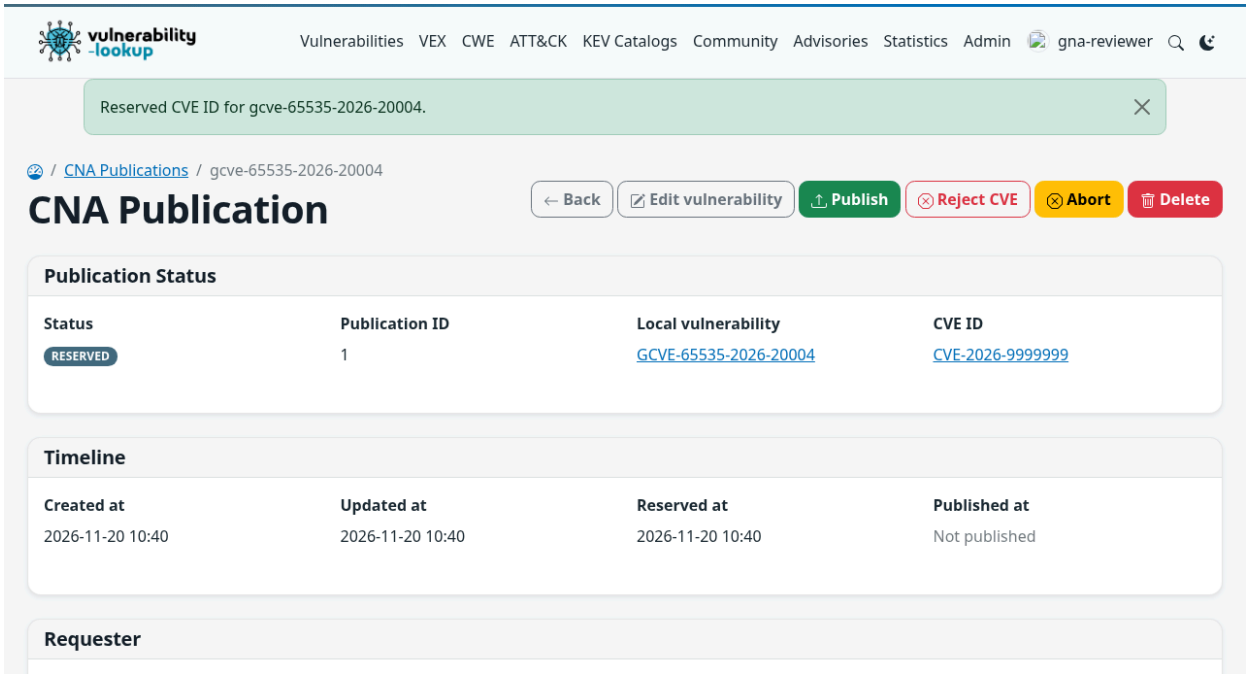


Figure 6.6: The publication details show the CVE Services reservation beside the local GCVE ID.

6.7 Publish, verify, and maintain the record

1. At the agreed release time, verify the chosen record, CVE ID, and advisory URL, then click **Publish** on the CNA publication. Treat `PUBLISHED` as acknowledgement of the submission. Check the official CVE record after distribution and compare the public product scope, fix guidance, credit, and references. CVE Record search
2. If the operation remains `RESERVATION_PENDING`, `PUBLICATION_PENDING`, or `REJECTION_PENDING`, check the publication's error and audit trail. For ambiguous timeouts, check the outcome in CVE Services before retrying the existing publication. This matters especially for reservations because the upstream service may already have allocated an ID.
3. Publish the user-facing advisory at the agreed time. In **Review**, use **Vulnerability ID** for the case's primary public identifier and add the other verified identifiers to the public advisory when they apply. Check the public text and set the case to **Disclosed**. Verify the case and advisory while signed out. Advisory and follow-up

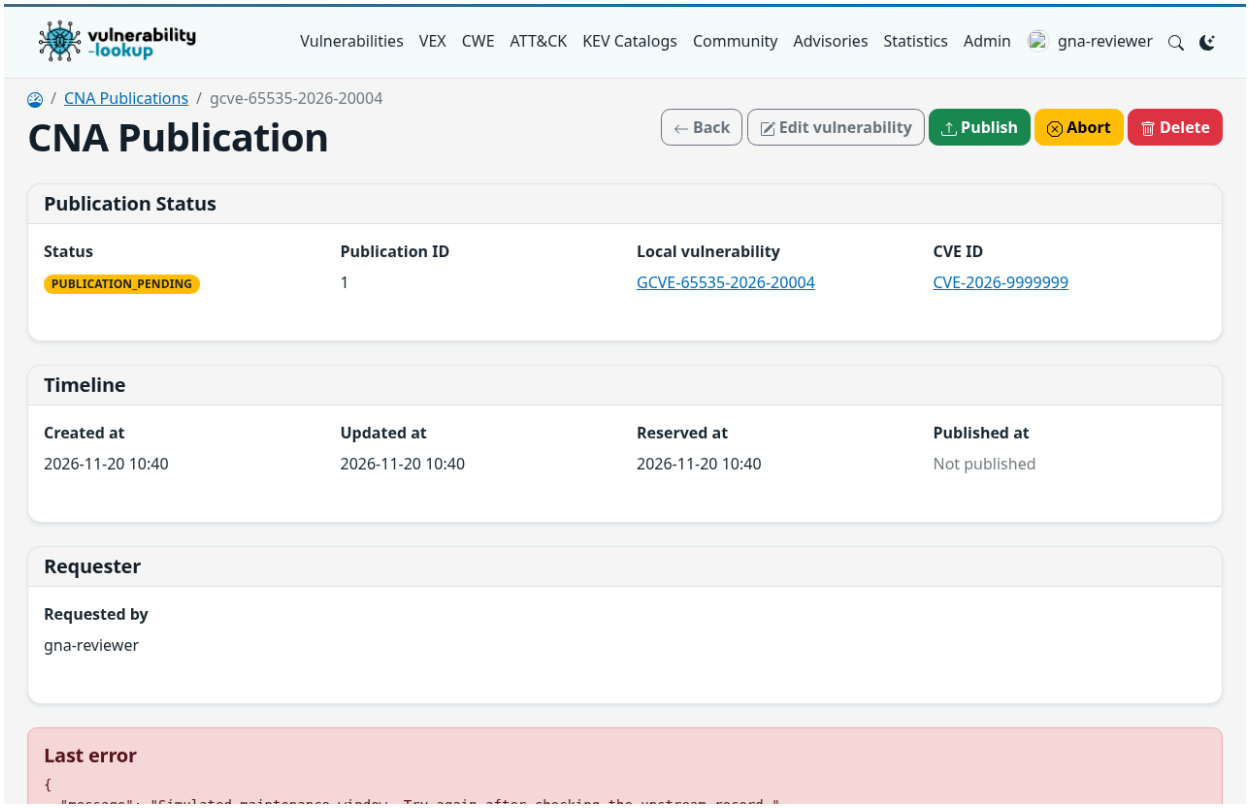


Figure 6.7: An upstream failure leaves the publication pending.

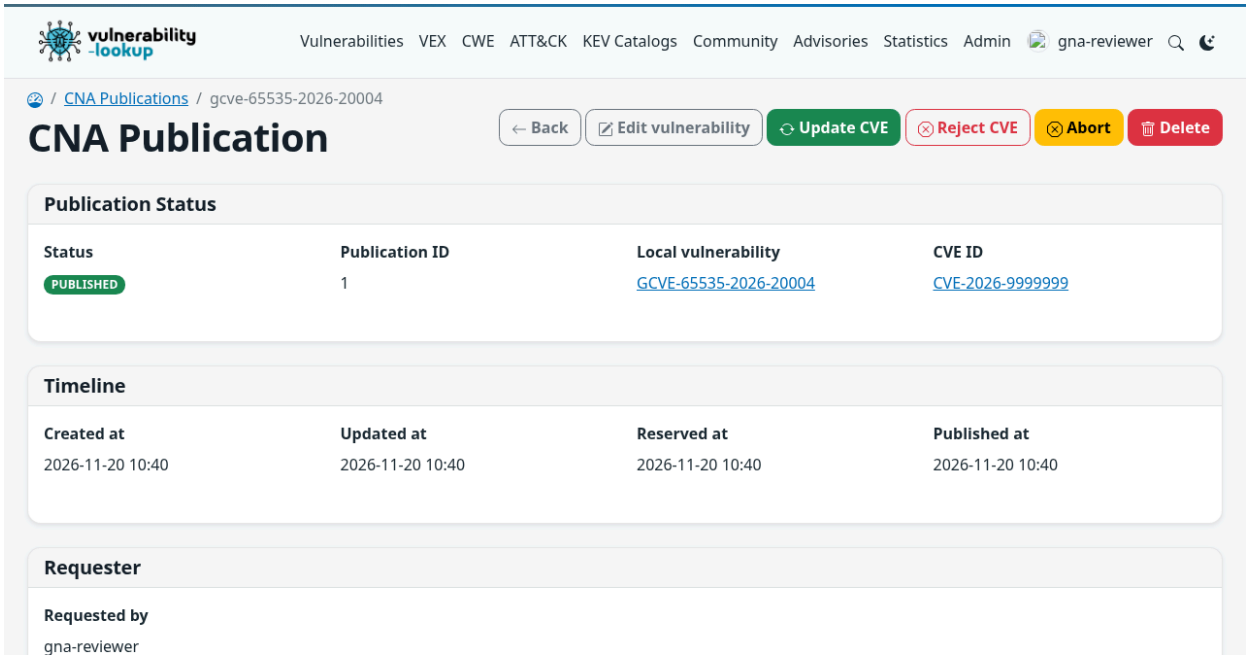


Figure 6.8: After the retry, the publication is marked Published and Update CVE is available.

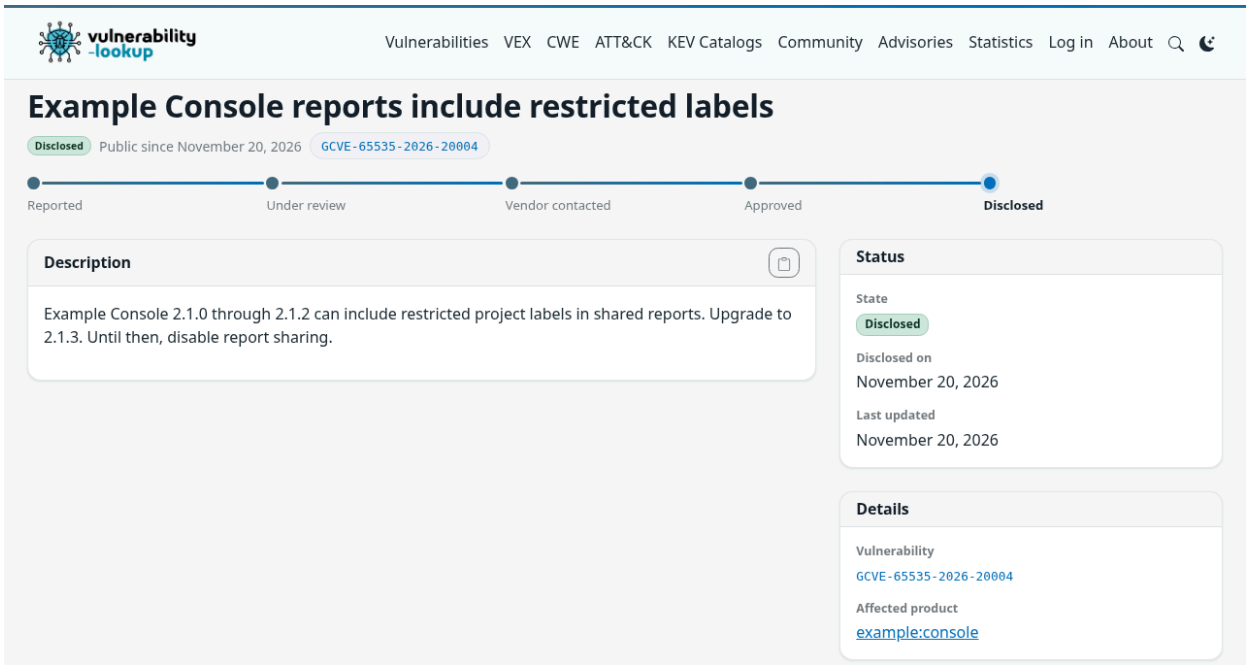


Figure 6.9: The public case links to its primary identifier.

For corrections, edit and save the local record, then use **Update CVE** on the existing publication. Verify both destinations, including the GNA output if it is also published. For an ID that must be withdrawn under CVE policy, use **Reject CVE** with an approved reason and verify the upstream result. **Abort** updates the local workflow only. Use `/publications/audit` on your Vulnerability-Lookup instance to review attempts and handoffs.

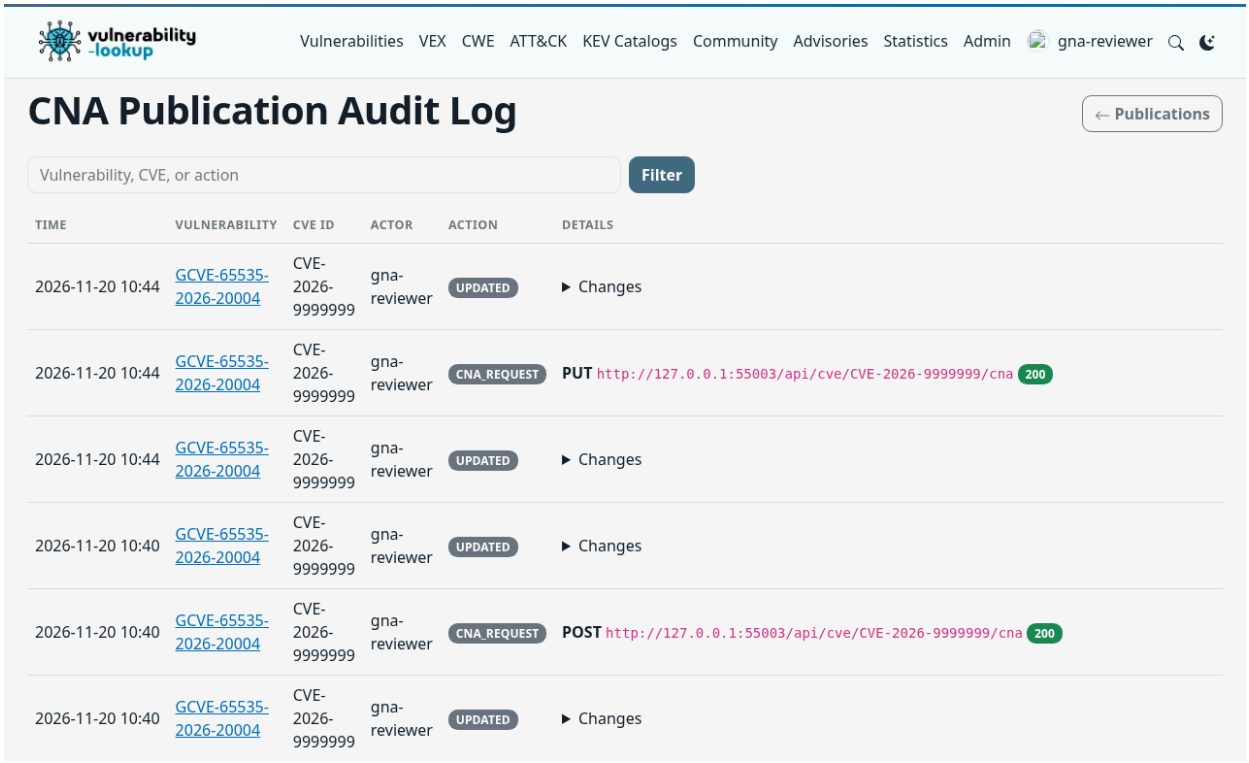


Figure 6.10: The publication audit tracks retries and later updates to the same publication.

6.8 Coordinate CNA and GNA publication on one instance

An organisation that operates both a CNA and a GNA can configure one Vulnerability-Lookup instance with its GNA local source and its CNA CVE Services settings. The same private disclosure and saved source record can support a local GCVE publication and a tracked CNA submission. This lets teams prepare scope,

remediation, references, and credit once, then manage each ecosystem's release and follow-up in the same instance. Each instance has one configured CNA short name and one local source identity. Use separate instances for separate publishing organisations. One organisation per instance

Coordinate	What to verify
Scope and identifiers	Decide which ecosystem outputs are needed. Record one primary identifier in the disclosure case and keep both identifiers in the release plan. Use the common advisory URL in both records. When linking a GCVE record to a CVE, use a GCVE relationship that reflects the scopes. Use `equal` only if they describe the same vulnerability.
Release order	Treat saving the local record, reserving the CVE ID, submitting to CVE Services, and disclosing the case as separate actions. A CVE reservation can appear in the GNA feed before the official CVE is published because the service mirrors the ID into the local record. Check both channels independently.
Corrections and status	Save the local GCVE correction, submit **Update CVE** for the tracked CVE as applicable, and update the advisory. Verify the GNA publication feed and official CVE record separately. A local save does not resubmit the CVE, and an upstream publication or rejection does not disclose or correct the CVD case.

GCVE record relationships · GNA release and verification

CHAPTER 07

Publish as a GNA

A GCVE Numbering Authority (GNA) assigns identifiers in its own namespace and publishes records from its own instance. This chapter follows a report through Vulnerability-Lookup, from private intake to a public GCVE record and advisory. It covers the GCVE BCP-02 handling lifecycle, with BCP-04 for identifiers, BCP-05 for record format, and BCP-03 for distribution.

The walkthrough uses the Vulniverse editor in Vulnerability-Lookup 7.1.0 development revision `b4c82c1`. All screenshots show a local instance, fictional report data, and GNA 65535, the GCVE test namespace. Use your assigned GNA number for production.

7.1 Become a GNA

Teams can publish advisories under their own local identifiers in Vulnerability-Lookup while preparing a GNA application, or keep local publication as their long-term approach. Apply for a GNA number when your organisation needs to allocate identifiers in an official GCVE namespace. We recommend that eligible organisations and vendors apply for both CNA and GNA status so they can choose the publication route that fits each case. Start with local publication · Become a CNA

1. **Check your eligibility.** An existing CVE Program CNA can apply. Other eligible applicants include CSIRTs or CERTs listed by FIRST, in the EU CSIRTs Network, or in TF-CSIRT. Software, hardware and service providers can apply if they regularly disclose vulnerabilities affecting their own products or services and have an official CPE vendor name. An organisation can also qualify with a public disclosure policy and a public source that publishes new vulnerability records in GCVE-BCP-05 format. GCVE FAQ on GNA eligibility
2. **Request a GNA number.** Email gna@gcve.eu with your organisation's name, the eligibility criterion that applies, and relevant directory information. Include your preferred short name, disclosure policy or advisory URLs, and applicable metadata such as a CPE vendor name or machine-readable publication endpoint. GCVE application guidance · Directory fields
3. **Confirm the assignment.** Verify your allocated GNA number and organisation details in the GCVE directory. Configure the GNA namespace and confirm the publication endpoint for discovery before issuing production GCVE IDs. GNA directory · Decentralized Publication Standard

7.2 The workflow

A disclosure case holds the report, discussions, and the team's release decision. An ID reservation allocates a number for that case. The GCVE record describes the issue in a machine-readable format, and the advisory explains what affected users should do. Coordinate their release times and verify each public destination independently.

If your organisation also operates a CNA, Vulnerability-Lookup can manage both publication routes from one disclosure case and local source record. Configure the instance for your GNA identity and enable the CNA Publication Service with your authorised CVE Services settings. Track the CVE submission separately from the local GCVE record. Coordinate CNA and GNA publication on one instance

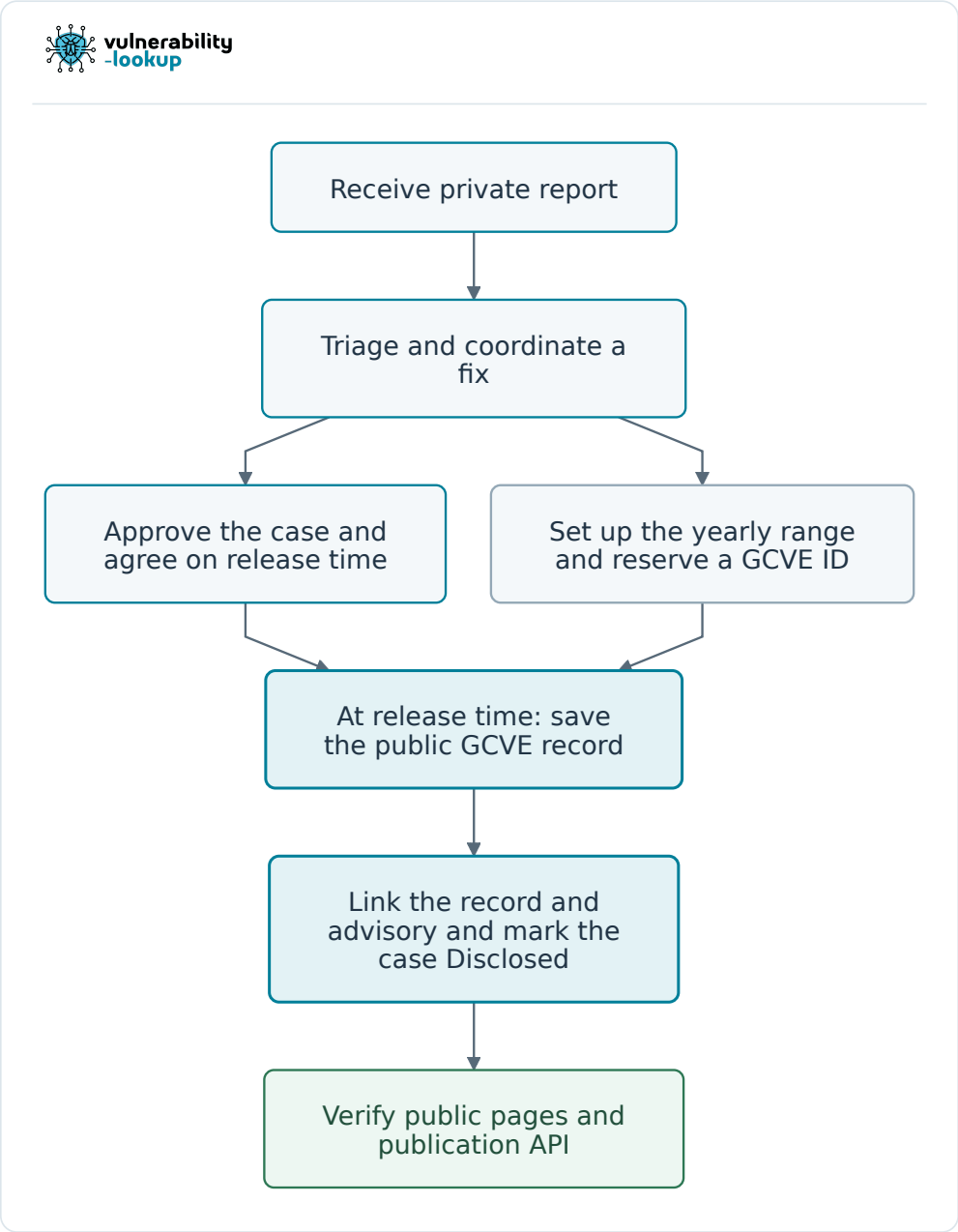


Figure 7.1: The GNA workflow, from report intake to public release and verification.

7.3 Set up the GNA

1. **Confirm your GNA number.** Use the recommended format `GCVE-<GNA number>-<year>-<serial>` with Vulnerability-Lookup's yearly allocator. Confirm the scope in which your organisation will assign IDs and who can approve releases. Become a GNA · GCVE ID allocation
2. **Choose a deployment.** The minimal publication guide covers PostgreSQL, Valkey, Kvrocks, and the web service. A publication-only node can leave all feeders disabled. If you need imported sources for correlation, enable the feeds you want on that instance or perform duplicate checks on a separate lookup with those feeds.
3. **Set the instance identity.** In `config/generic.json`, enable `user_accounts`. Set `local_instance_name` to `gna-<your GNA number>` and set `local_instance_vulnid_pattern` and `local_instance_vulnid_example` to match the namespace and serial format. Generate a unique `local_instance_uuid` once and retain it across restores. Set `public_domain` to your externally reachable HTTPS hostname. Review the upper serial limit and priority boundary before creating the year's range.
4. **Enable the operator tools.** In `config/website.py`, turn on `vulnerability_disclosure` and `vulnerability_editor` in `WEB_MODULES` and set `VULNERABILITY_EDITOR =`

`"vulniverse"`. Keep the `gcve-reservation` panel enabled in `config/vulniverse.json`. Set `ADMIN_EMAIL` to the coordination mailbox and `CVD_POLICY_URL` to your public disclosure policy. Apply migrations, restart the service, and verify mail delivery before taking real reports. Disclosure configuration · Vulnerability editor

5. **Plan verification and recovery.** Ask the GCVE directory maintainers to register your publication base URL in `gcve_pull_api`, then test the endpoint from outside the instance. If you use a local copy of the directory for GNA metadata, refresh it with `poetry run flask --app website.app update_gcve_registry`. Back up the configuration, PostgreSQL case data, and Kvrocks records and allocations together. Rehearse restoring all three before relying on the backups. GCVE publication transport

7.3.1 Grant operator permissions

Use **Admin → Roles** to create roles for the people who perform each action, then assign them to the relevant accounts. Test the roles with separate accounts on staging. The review queue and **Review** form are admin surfaces in this release. Coordinators with `vulnerability_disclosure:review` can open known cases and use **Discussion** and **Notes** without full admin access. Roles and permissions · Disclosure roles

Task	Permission or access
Submit a report	`vulnerability_disclosure:report`. The case-creation policy assigns owner access to the submitter.
Coordinate the case	`vulnerability_disclosure:review`. Use an administrator account for the admin queue and review form.
Create the annual range	`vulnerability_id_range:create`
Reserve a GCVE ID	`vulnerability_id:reserve`
Publish or correct a local record	`vulnerability:create` for publication and `vulnerability:modify` with access to the record for updates.

For external submitters, use a dedicated role with the report permission. In this release, the built-in `reporter` role also grants local vulnerability record permissions. Check the **Self-Registration Policy** and account confirmation flow before offering public registration.

7.4 Receive and coordinate a report

1. Have the reporter sign in and open `/disclosure/new/`. Ask for the affected product, versions, impact, and a safe summary. Put sensitive details in **Confidential details**. The main **Description** can become public. Ask whether the reporter wants credit and verify the choice at release.
2. In **Admin → Disclosures** (`/admin/disclosures/`), open the new **Reported** case. Assign a case owner, check scope and existing identifiers, and use **Review** to move the case to **Under review**. Acknowledge receipt in **Discussion** and give the reporter a date for the next update.

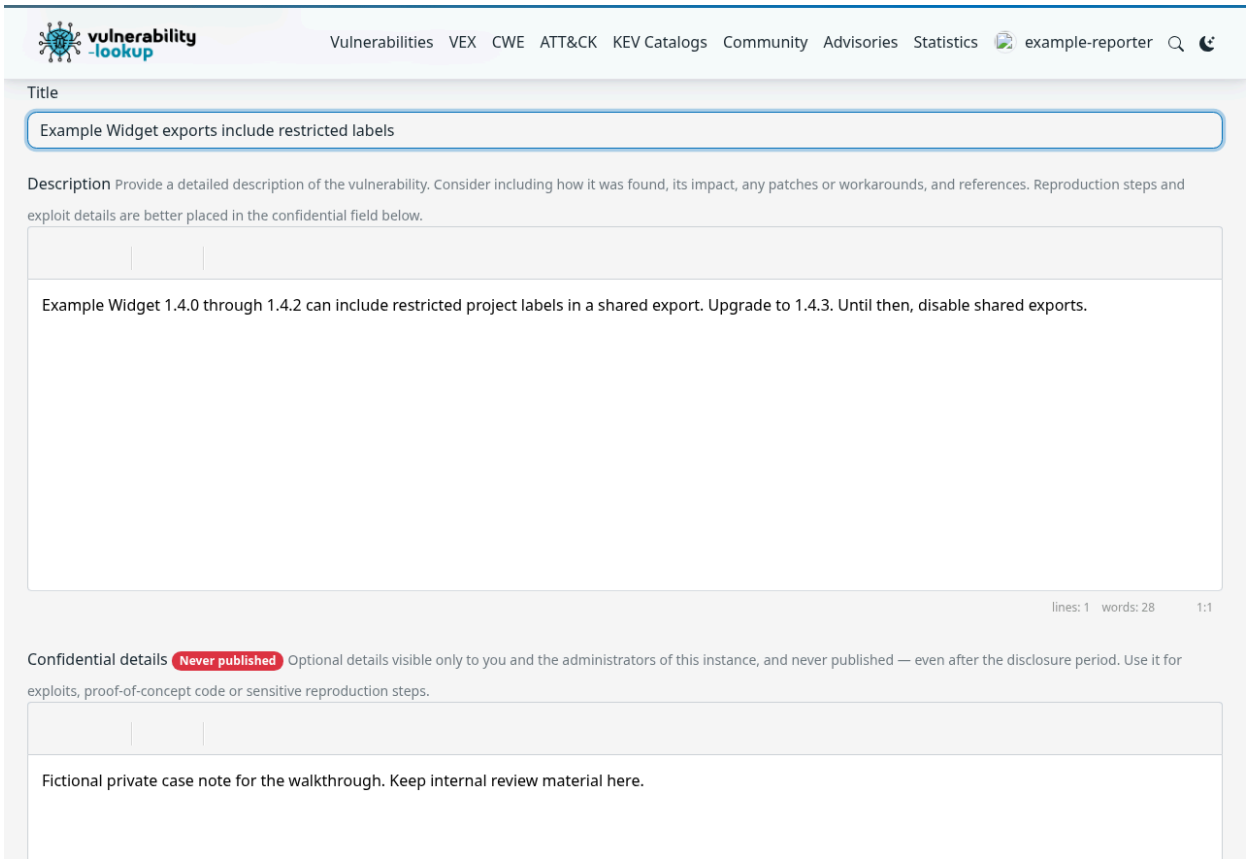


Figure 7.2: The report form separates the public-facing description from confidential details.

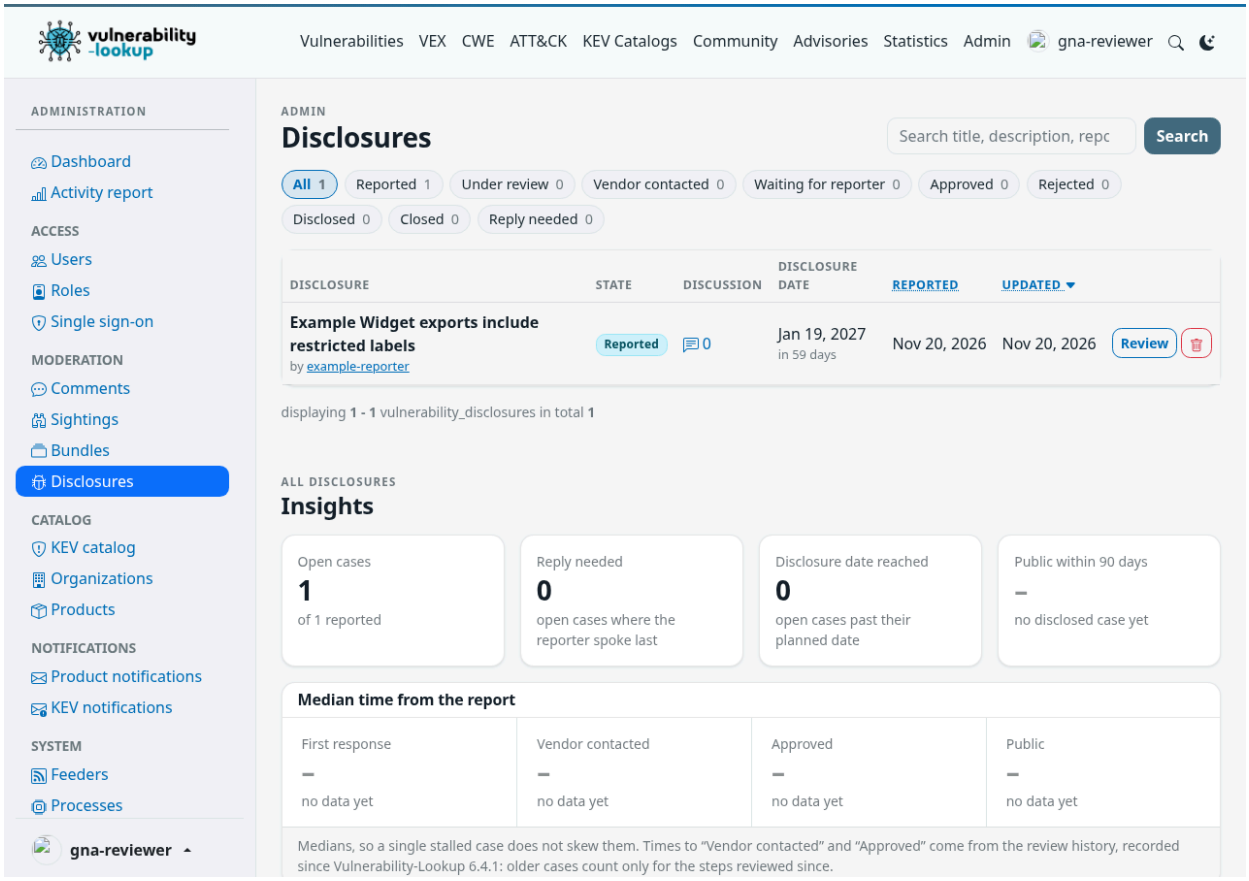


Figure 7.3: The coordinator opens a newly reported case from the admin queue.

3. If more information is needed, use **Waiting for reporter**. After contacting the maintainer, use **Vendor contacted**. Keep reporter-facing messages in **Discussion** and private decisions in **Notes**. Use **External reference** on the review form for the restricted coordination ticket.

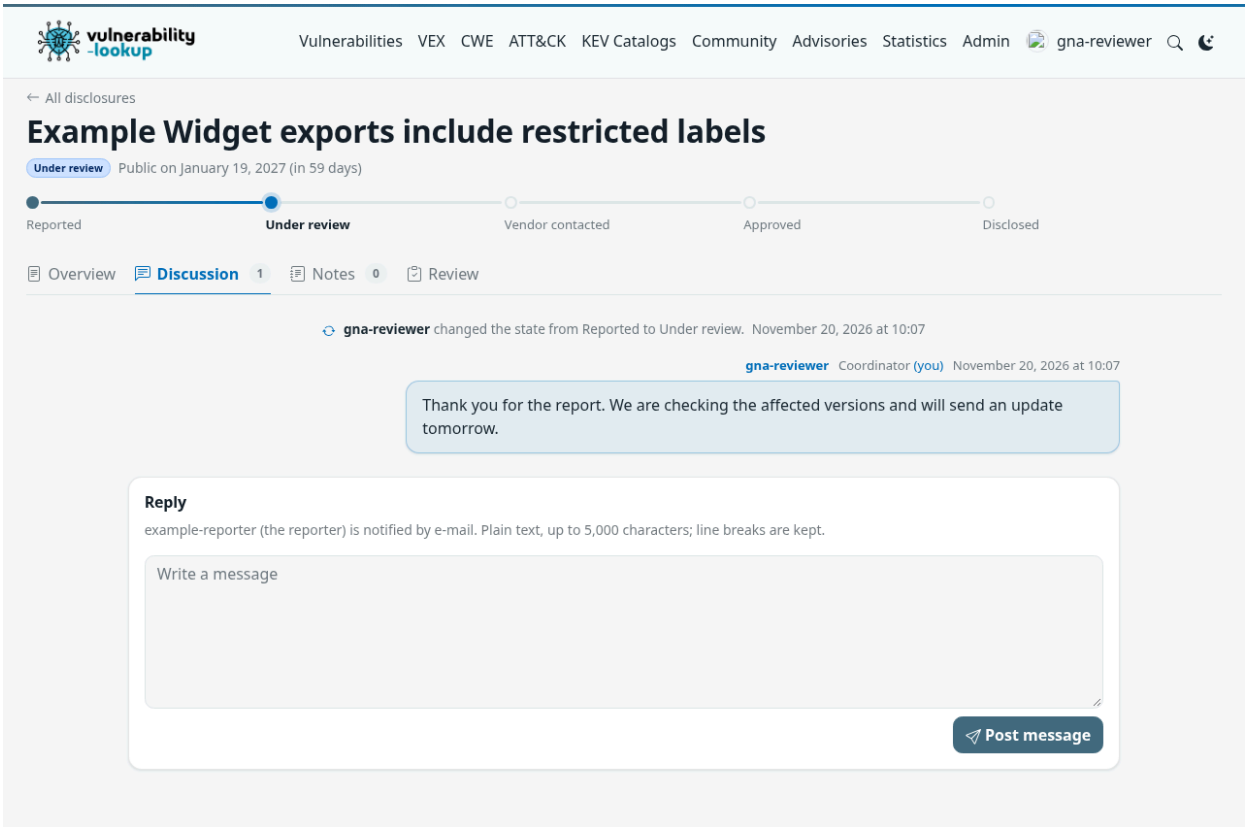


Figure 7.4: The case discussion shows the acknowledgement sent to the reporter.

4. Confirm affected and fixed versions, mitigations, and release timing with the maintainer. Record the release time and time zone in the coordination plan, and set the corresponding **Disclosure date**. Move the case to **Approved** once the release is agreed. That state and date leave the case private until an operator explicitly sets it to **Disclosed**. Investigation and coordination

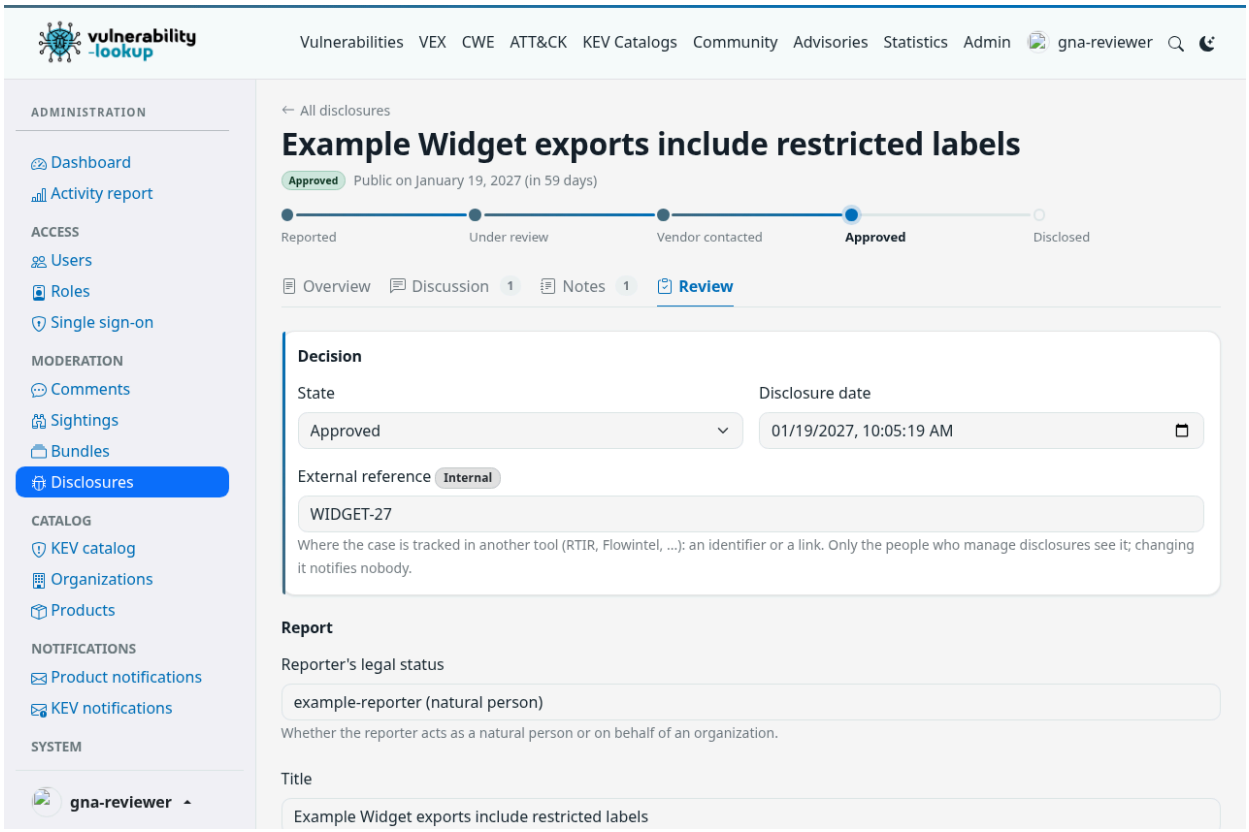


Figure 7.5: The review decision records the status, target disclosure date, and internal coordination reference.

7.5 Create the yearly range and reserve an ID

1. Under **Advisories → New vulnerability advisory**, open Vulniverse's **GCVE ID** panel. Expand **Create range document**, enter the allocation year, and create the range. Do this once per year before issuing IDs for it. If the range already exists, use it. Preserve the existing allocations when moving or restoring the instance.
2. Expand **Reserve new ID**, enter the year, and request one ID. The editor places the first returned value in **GCVE identifier**. Record it in the private case or coordination ticket and use it in the advisory draft.

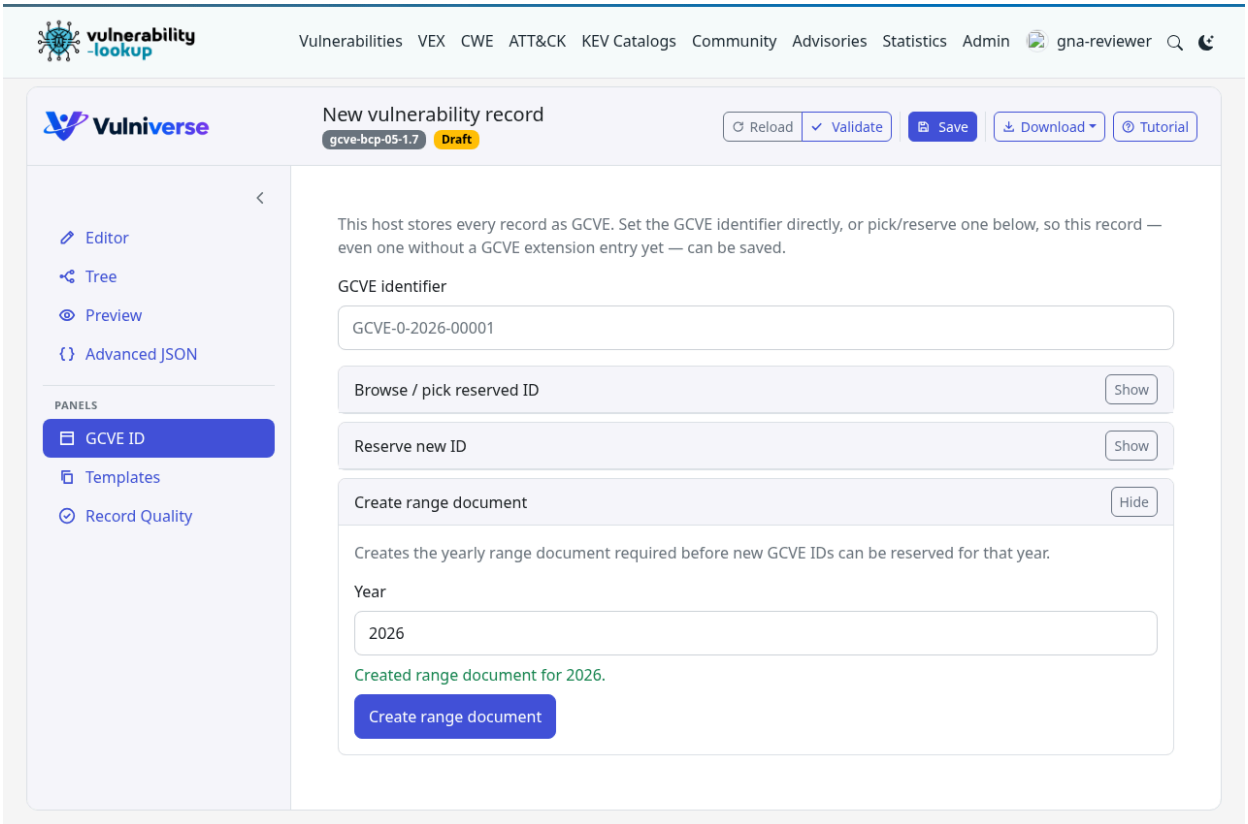


Figure 7.6: Vulniverse confirms that the yearly allocation range was created.

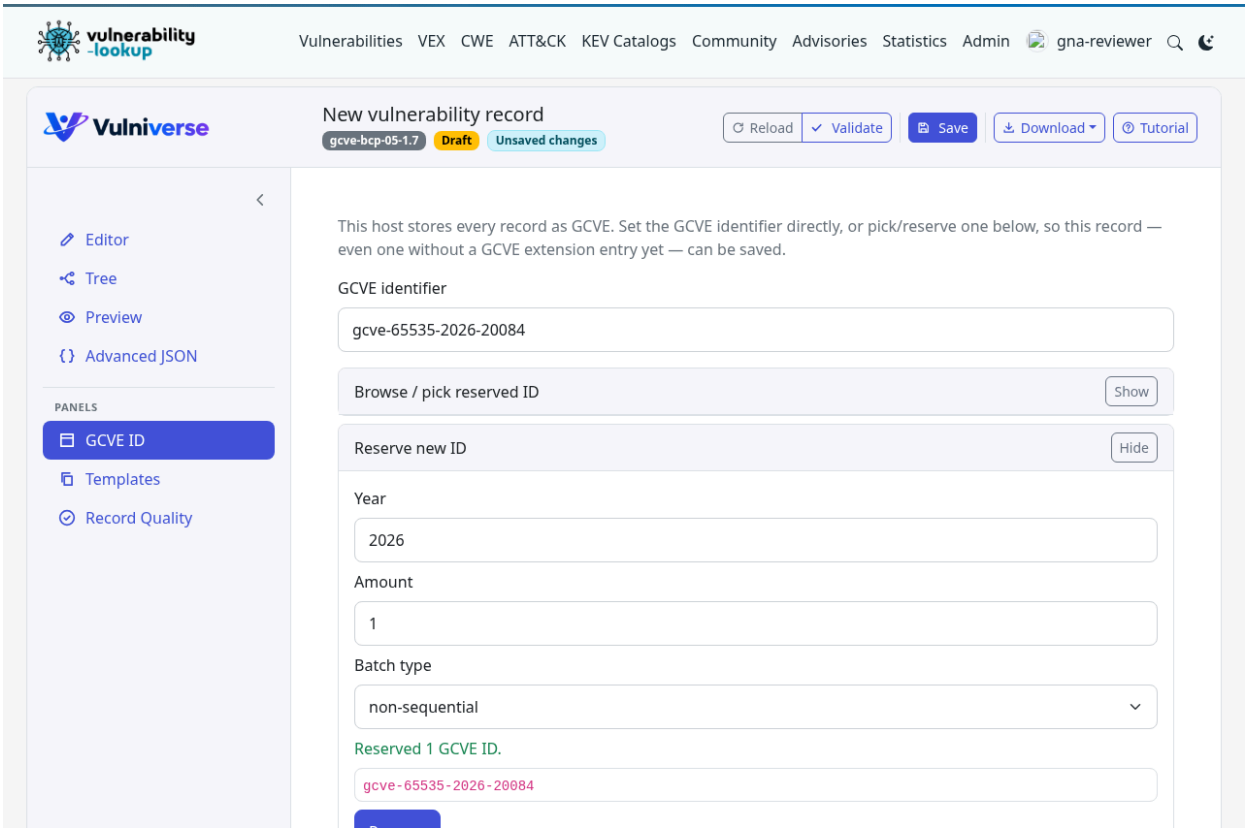


Figure 7.7: The GCVE panel reserves an ID and fills the record's identifier field.

Use **Browse / pick reserved ID** to reopen an allocation by identifier, year, or state. Choose the case's existing ID when resuming work after a handoff. If a reservation request times out, check the list before retrying so the case keeps one allocation. Keep confirmed identifiers from other sources in the record's

cross-references. If scope changes or a case closes without release, document the decision and retain the assignment history. GCVE ID allocation

7.6 Prepare, validate, and publish the record

Prepare the text in the restricted workspace while the case is embargoed. In Vulniverse, select the reserved ID, then enter the public title and description in **Summary**, the version ranges in **Affected products**, the fix and mitigations in **Guidance**, and the advisory URL in **References**. Add approved credit under **Credits and timeline** and use the **GCVE extension** for record relationships where applicable. Check **Preview** against the release checklist, then click **Validate** to check the record format. Write and evaluate a vulnerability record · GCVE vulnerability format

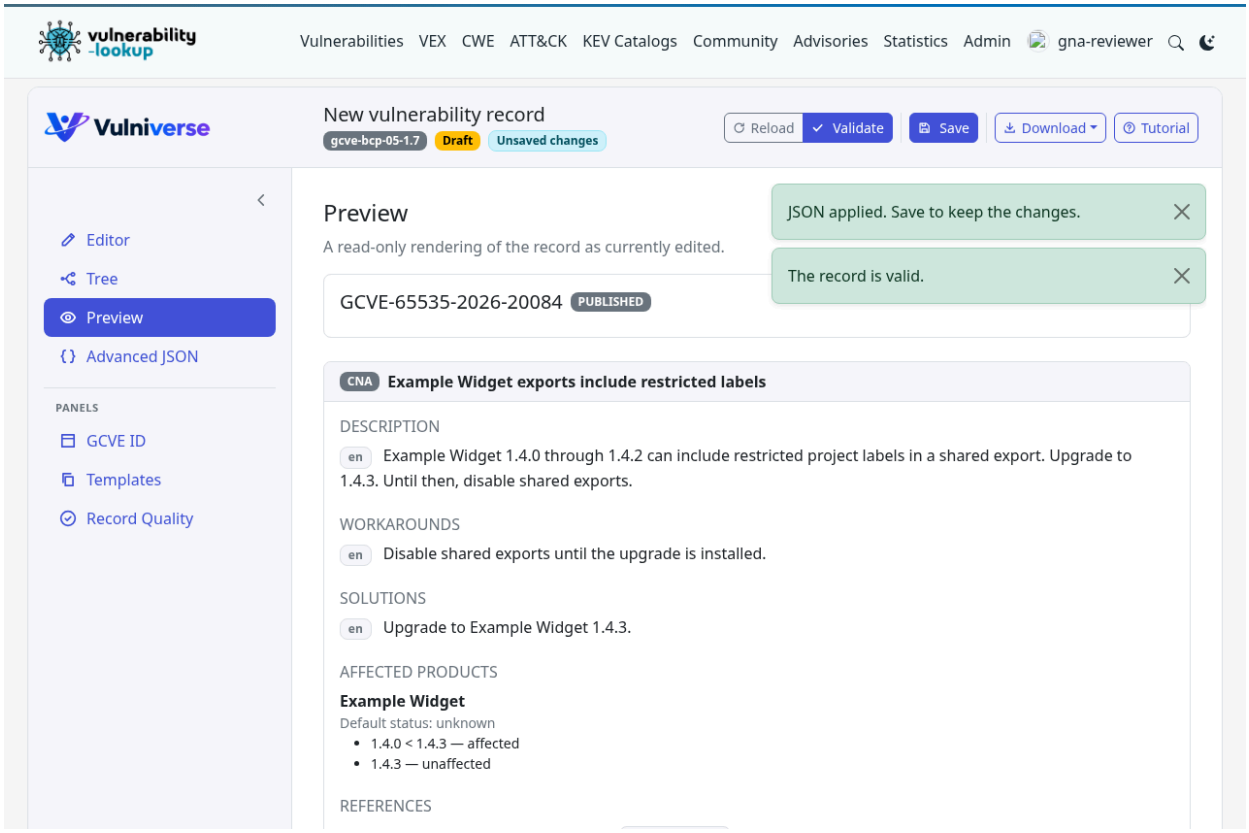


Figure 7.8: In Vulniverse, review the approved record and validate it before saving.

> Save at the agreed release time
> On this Vulnerability-Lookup revision, **Save** publishes a new GCVE record on the local source. The **Draft** badge indicates an unsaved record in the editor. **Preview** can show the proposed record state as **Published** before that save. The disclosure case's **Approved** state controls only the case page. Use the signed-out API to verify whether the record is public.

At release time, save the record and open its public page. Publish the user-facing advisory through the agreed channel and verify the links in both directions. In the case's **Review** form, enter the published GCVE ID under **Vulnerability ID**, verify which report fields will be visible, and set **State** to **Disclosed**. Independently record the time each publication step succeeded. Advisory and follow-up

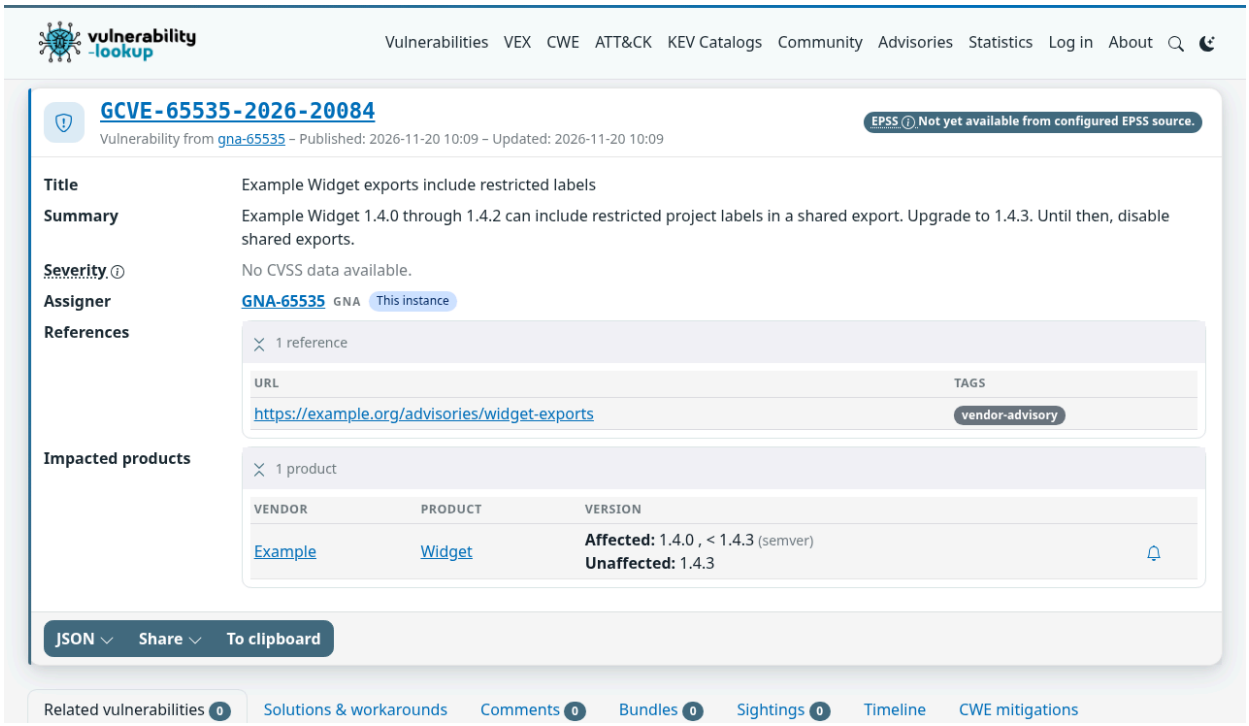


Figure 7.9: The signed-out vulnerability page shows the GCVE identifier, version ranges, and advisory reference.

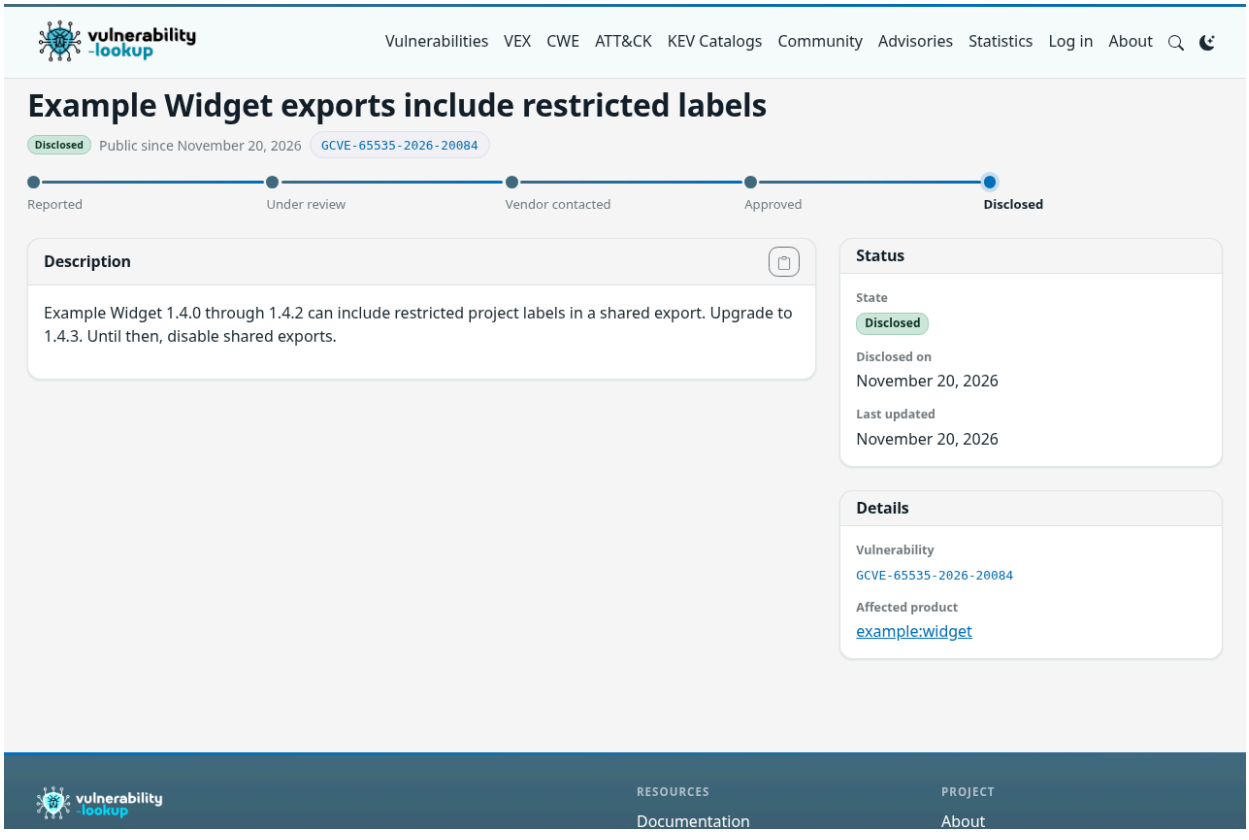


Figure 7.10: The public disclosure links back to the published vulnerability record.

7.7 Verify and maintain your public source

- **Check the web views while signed out.** Open the GCVE record, its advisory reference, and the disclosure from `/disclosures/`. Match the ID, dates, affected versions, fix guidance, and credit. Check that confidential details, private notes, and internal references are absent.
- **Check collection over HTTP.** Query `GET /api/gcve/publication` on the published host and confirm the local record and its identifier are present. If you offer a static dump, check that `/dumps/`

`gna-<number>.ndjson` includes the record after export and that the export is refreshed on your publication schedule. Use the documented publication transport and the Vulnerability-Lookup dump commands.

For a recent release, set the instance URL and identifier, then look for a matching entry on the first page of the public JSON feed. Page through the feed or narrow the date window for older records.

```
base_url="https://vuln.example.org"
record_id="GCVE-65535-2026-20084"
curl -fsS "$base_url/api/gcve/publication?per_page=100" |
  jq -e --arg id "$record_id" '
    .[] | select(any(.containers.cna.x_gcve[]?; .vulnId == $id))
    | .cveMetadata | {state, datePublished, dateUpdated}
  '
```

- **Check discoverability.** Compare your directory entry and configured source with the URLs that consumers actually retrieve. If the source already serves the record, investigate the consumer's fetch schedule or filters before taking further publication action.
- **Correct the same record.** Open the published record and select **Edit**. Update it under the same identifier, verify the modified record via the publication API, and update the advisory with a dated correction. Inform the reporter and affected publishers where appropriate. Reopen coordination if the affected scope or available protection changes.
- **Review operations.** Watch for replies needed, disclosures nearing their target date, and inconsistent public records. Verify the remaining annual ID capacity, test backups of case and allocation data, and review delays after release. CVD insights · Release checklists

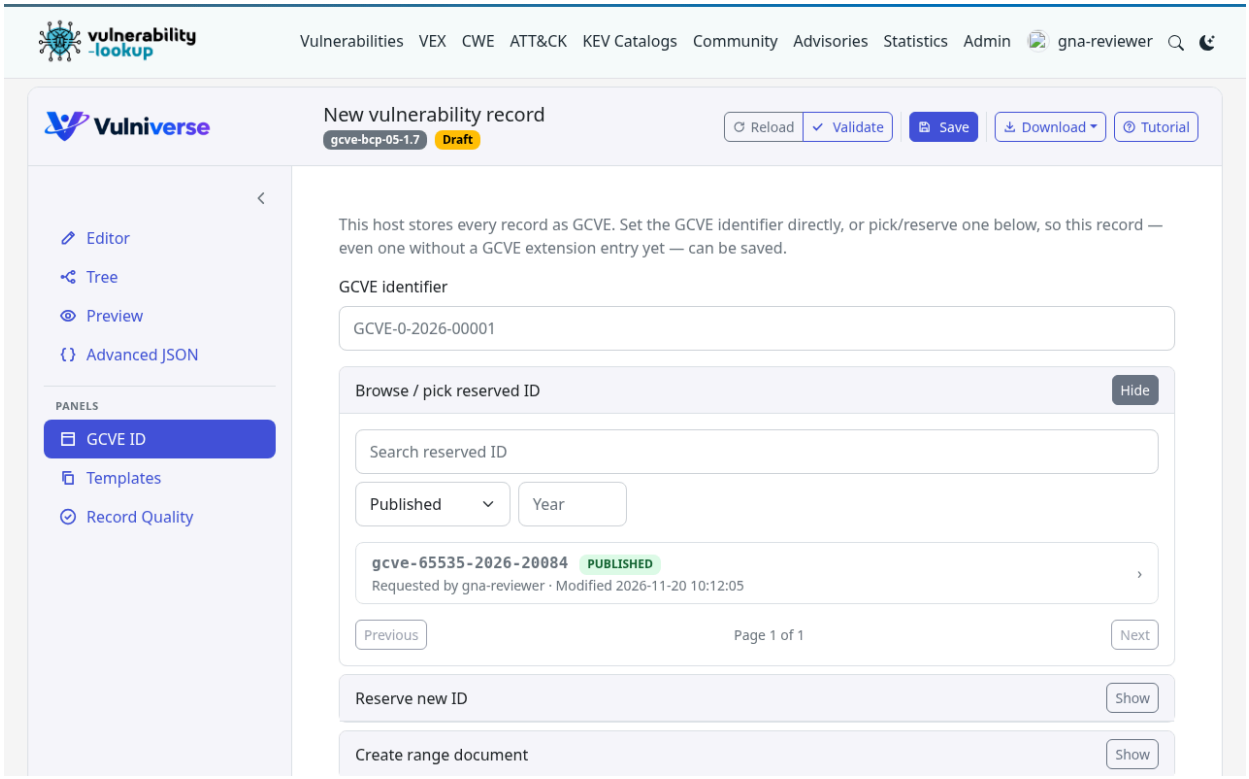


Figure 7.11: The GCVE ID panel can find an existing record when an operator needs to make a correction.

7.8 Keep the GNA and CNA outputs in sync

When a case uses both routes, keep the GCVE ID, CVE ID, and common advisory URL in the release plan. Use the disclosure's **Vulnerability ID** field for the primary public record and spell out both verified identifiers in the advisory. The GNA 0 form `GCVE-0-<year>-<serial>` represents the existing `CVE-<year>-<serial>`. Reserve in your own GNA namespace when you have a distinct publication to make. GNA 0 registry entry · GCVE ID allocation

On a dual-role instance, reserving the CVE through the CNA Publication Service attempts to mirror it into `cveMetadata.cveId` on the local record. The GNA API can expose that field while the CVE is still reserved upstream. Account for this in the release window, and check the official CVE record before describing it as published. For a correction, save the GCVE record and verify the GNA feed, then submit **Update CVE** through the existing CNA publication and verify the CVE separately. CNA publication and follow-up

CHAPTER 08

Publish the advisory and follow up

8.1 Release together

Review the release checklist with the coordinator and publication operator. Prepare an advisory that identifies the affected product and versions, explains user impact, and tells readers how to obtain and verify a fix or apply a mitigation. If a fix is pending, state what users can do now and where to find updates. Include the public identifiers, relevant references, and the credit agreed with the reporter.

1. Publish the public vulnerability record through the agreed route and verify the identifier. Link it in the public advisory and link the record back to the advisory. The Example Portal walkthrough uses the GCVE test namespace. Use your agreed local identifier, CNA, GNA or partner publication route in production. Local publication and CNA publication · GNA publication
2. On the case's **Review** tab, check the title, description, affected product, and credit settings. Enter the public **Vulnerability ID** and set **State** to **Disclosed**. Select **Save review** at the coordinated release time. The transition replaces the planned date with the actual disclosure time and the reporter receives a review notification. Publishing the vulnerability



Figure 8.1: The release operator enters the published identifier in the case before saving the review.

8.2 Verify as a reader

In a signed-out session, open the advisory, public record, and disclosure listing at `/disclosures/`. Check that the identifiers and version ranges agree, the update links work, and the public pages contain only approved details. If you distribute through a feed or API, verify the same record there. Give the reporter and participating teams the public links. Feed syndication · API

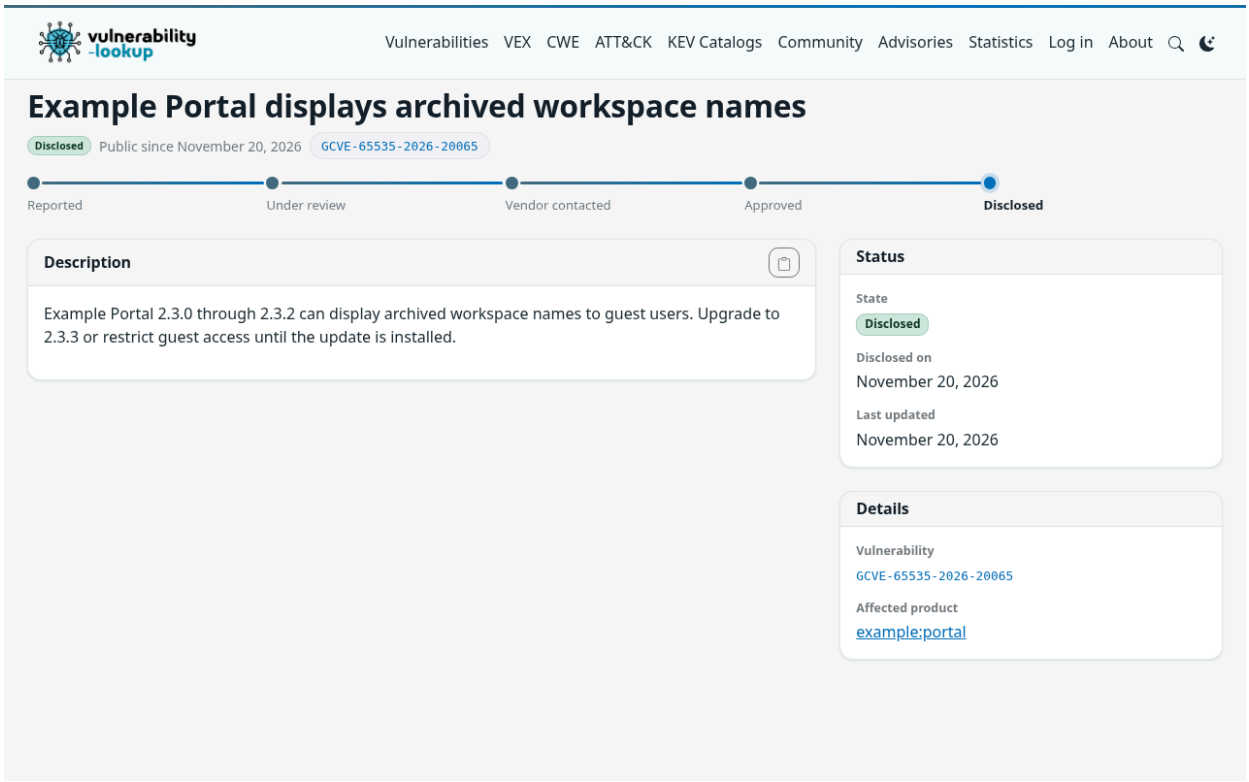


Figure 8.2: The signed-out disclosure links to the published vulnerability.

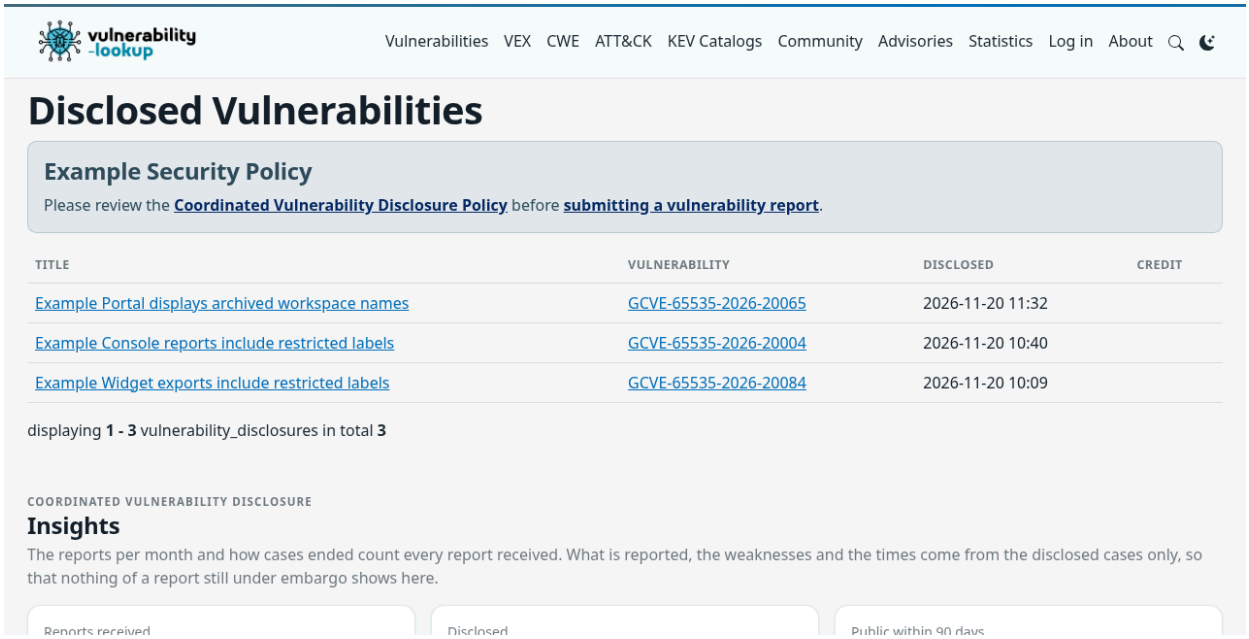


Figure 8.3: The case appears in the public disclosure list after release.

Follow the identifier link from the public disclosure to the vulnerability record. Confirm the agreed credit is displayed and that the reporter's account is hidden for an anonymous case. Check that the confidential details, discussion, notes, and external reference are restricted even after release.

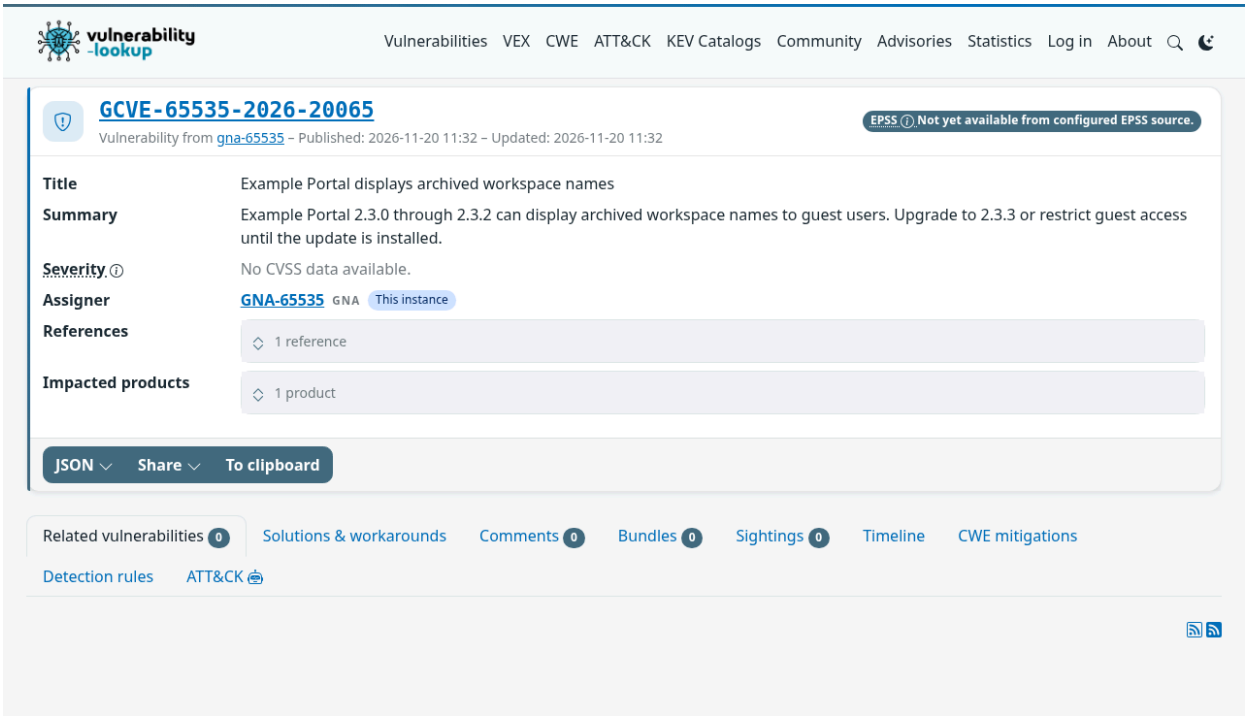


Figure 8.4: The public vulnerability record carries the same identifier and the remediation summary.

8.3 Keep the record current

Assign someone to handle feedback and corrections. If the affected scope or fix changes, update the advisory and the published vulnerability record, record what changed and when, and notify the other publishers. Route substantive new evidence back through triage. Review the case afterward and turn any missed handoffs into specific changes to the team's process.

Use **Insights** below **Admin → Disclosures** to review response and release times. The public **Insights** at `/disclosures/` shows aggregate counts across cases and statistics based on disclosed case content. The public figures refresh every ten minutes. Upcoming titles and private queue indicators remain in the admin view.

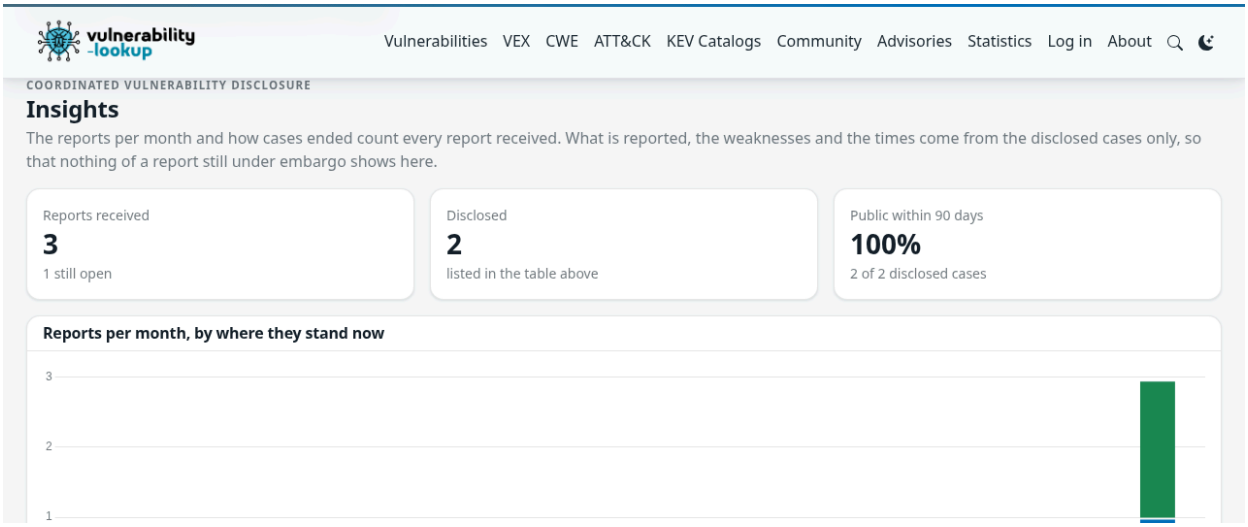


Figure 8.5: Public Insights shows aggregate disclosure activity.

When you verify exploitation and can share the finding, publish the evidence and affected scope as a separate assertion in your local KEV catalogue.

BCP-02 · Publishing Advisories

CHAPTER 09

Maintain a local KEV catalogue

Use the local Known Exploited Vulnerabilities (KEV) catalogue to publish what your organisation knows about exploitation. A vendor can attach a finding to its own advisory. A CSIRT can publish a finding about a vulnerability disclosed by another organisation. In either case, your instance publishes a separate assertion with its own status, scope, and evidence. The original advisory continues to describe the vulnerability and its remediation. You can run a catalogue without CNA or GNA status. GCVE BCP-07 assertion format

The screenshots use a local Vulnerability-Lookup 7.1.0 development instance. The vendor example follows the fictional Example Widget advisory from the GNA chapter. The CSIRT example uses a real CVE and public CISA evidence to rehearse the process without using a CRA notification. Publish as a GNA · Vulnerability-Lookup revision b4c82c1

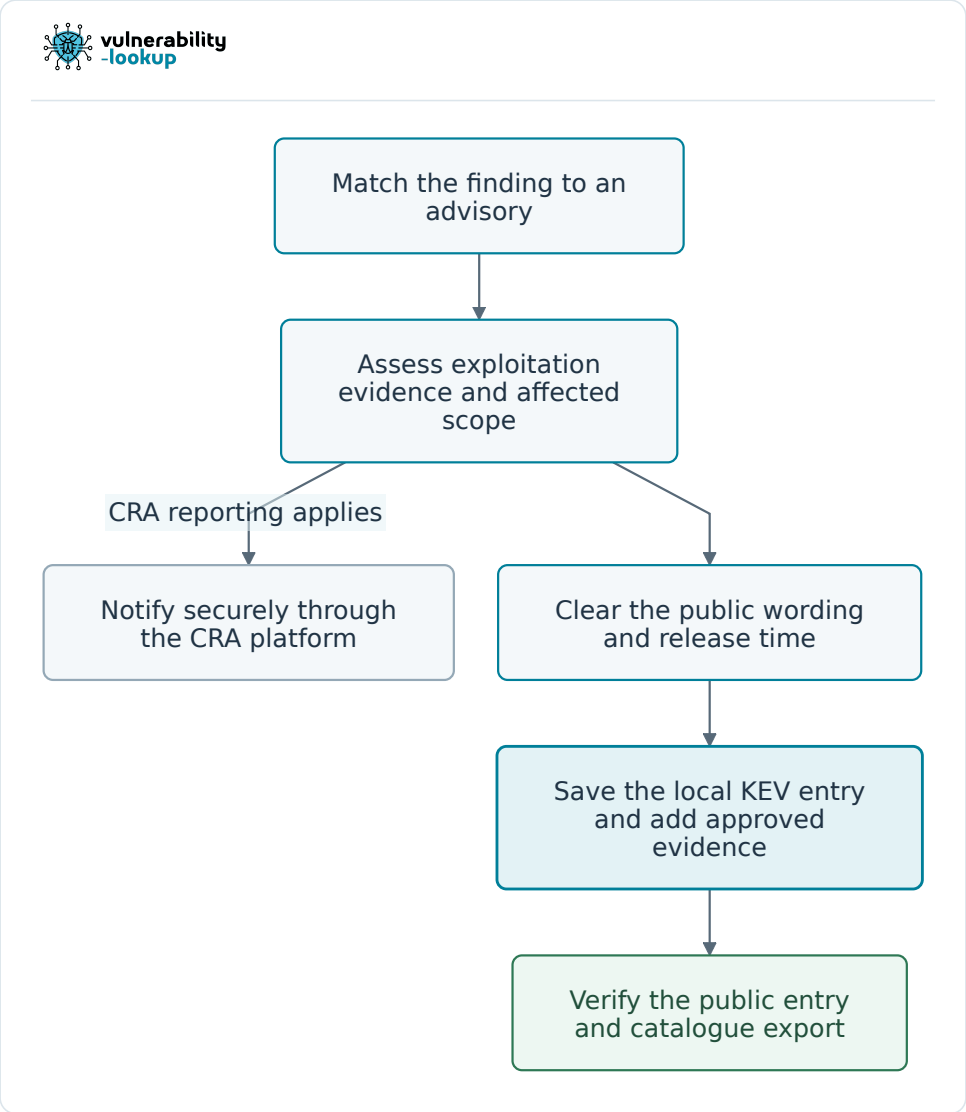


Figure 9.1: Review evidence and clear public wording before publishing to your catalogue. Where CRA reporting applies, run it on its own schedule.

9.1 Prepare the catalogue and assess the evidence

1. **Set the catalogue identity.** Give the instance a recognisable `local_instance_name` and a unique `local_instance_uuid` in `config/generic.json`. Preserve the UUID across migrations and restores so consumers can recognise the same catalogue. Use an account with `admin:access` for **Admin → KEV catalog**. The record-page shortcut also requires `vulnerability:modify`. Roles and permissions
2. **Match the public record.** Check the identifier, publisher, affected products, and versions against the evidence. Search the local catalogue by the primary and alternative IDs to find existing entries before creating a new one. Use alternative IDs only for records that identify the same vulnerability.
3. **Agree on what justifies listing.** Decide which observations or credible third-party reports support a positive assertion and who can approve one. Confirm exactly what the evidence shows, such as an attempt or successful exploitation. A CVSS score measures severity. Scanning, a proof of concept, or unverified attempts alone leave successful exploitation unconfirmed. Track unresolved observations in the restricted case until your team can review them.
4. **Approve the public wording and timing.** Prepare a concise source description, scope, and protection guidance. Keep raw telemetry, customer details, confidential reports, and internal case references in your restricted system. Saving the entry publishes its fields on a public instance. Adding evidence on the edit screen also makes that evidence available to readers.

9.1.1 Record the status and dates

Field	What to record
Vulnerability ID and **Alternative IDs**	Enter the primary public identifier and any verified equivalents, one per line.
Exploited and **Status Reason**	For a verified positive assertion, select **Exploited** and **Confirmed**. State what the evidence confirms in the scope or evidence description.
Status Updated At	Record when the catalogue decision took effect or changed.
First Seen At and **Last Seen At**	Use supported observation times for exploitation. Clear default values if the observation time is unknown.
Asserted At	Record when your organisation made the assertion.
Scope Notes	Describe the limits of the observation, relevant products or versions, and what users can do. Link to the advisory for the full affected range and remediation.

Review any timestamps prefilled by the form. Keep the observation time, catalogue decision, and the manufacturer's regulatory awareness time distinct. Leave optional characteristics and evidence fields empty when you lack a basis for them. GCVE BCP-07 status and timestamps

9.2 List exploitation of your own advisory

In this exercise, a vendor's incident responders confirm exploitation of Example Widget after its advisory is public. The vendor approves a summary of the finding and refers users to the fixed release.

1. Open your published record. Confirm its identifier and affected range, then select **Create KEV Entry**. The form takes the identifier from the record.

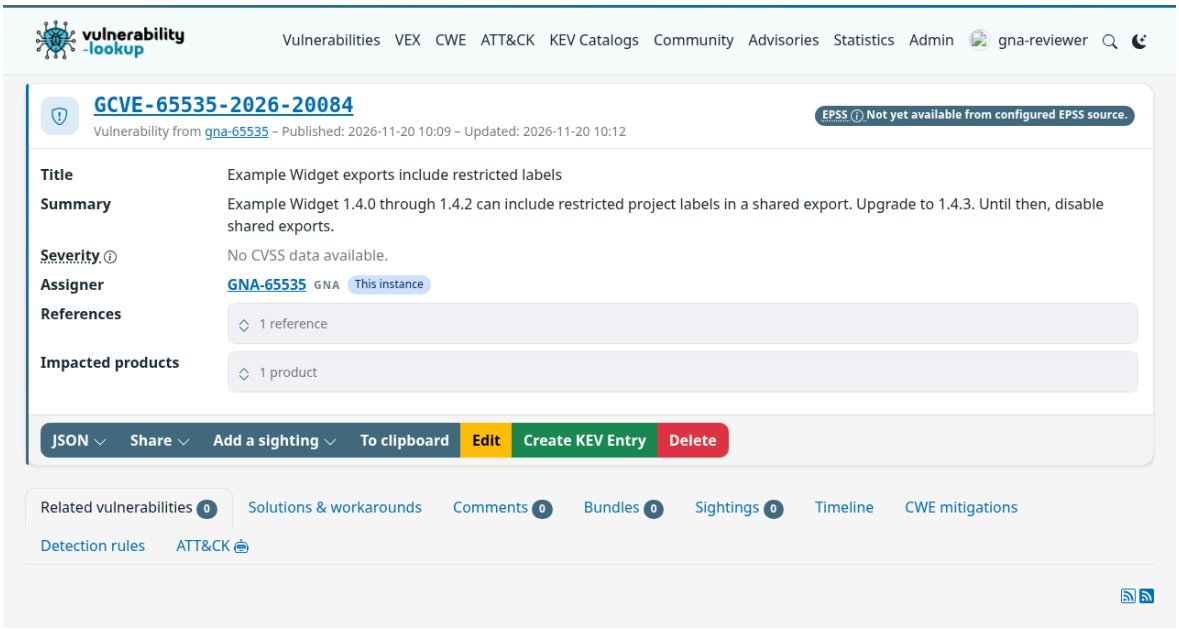


Figure 9.2: Open the local advisory and select Create KEV Entry.

2. Verify the prefilled **Vulnerability ID** and add your advisory ID in **Alternative IDs** if it identifies the same issue. Select **Exploited** and **Confirmed**. Check the dates, fill **Scope Notes**, and select **Save** once the assertion is ready to be public.

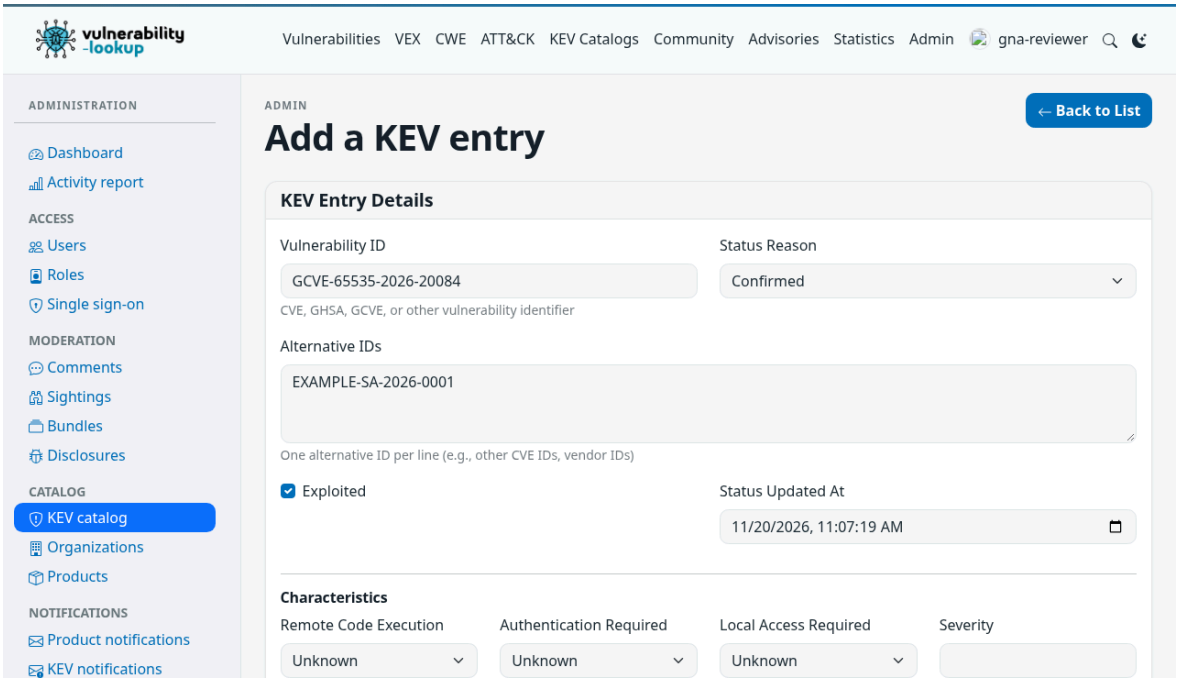


Figure 9.3: Set the status for the Example Widget assertion and review its dates.

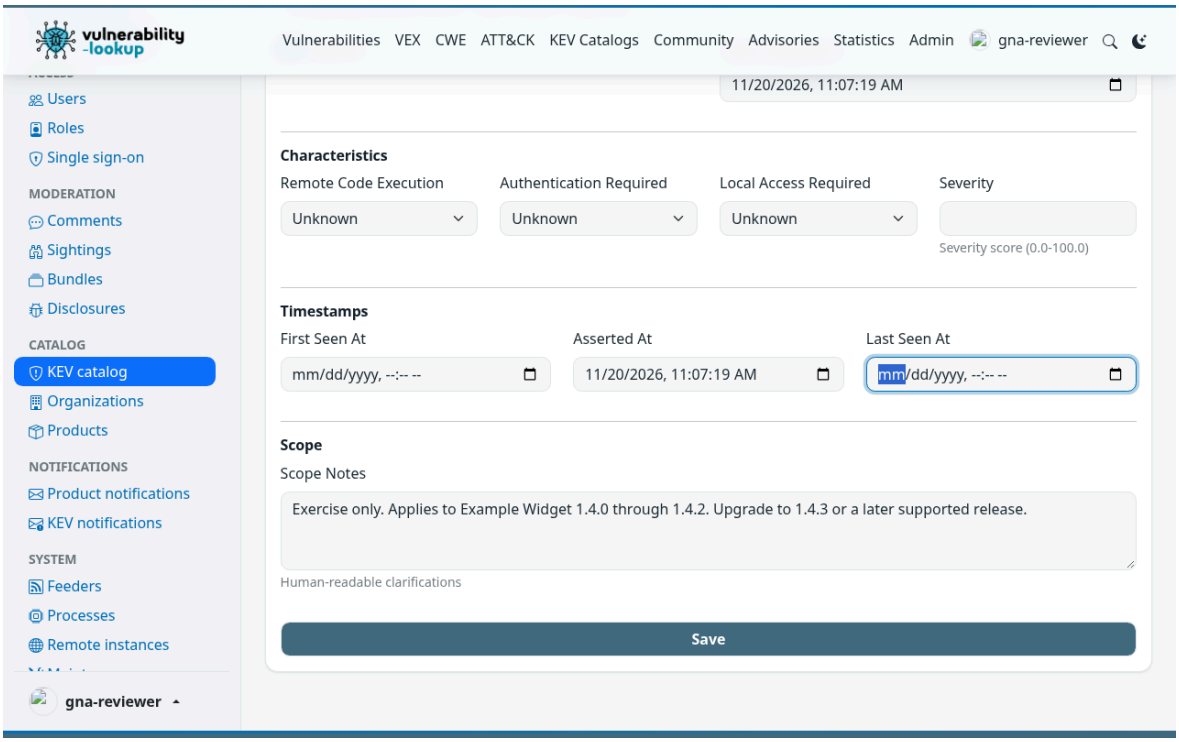


Figure 9.4: The vendor limits the exercise assertion to versions 1.4.0 through 1.4.2 and points to the fixed release.

3. On the edit page that opens after saving, find **Add Evidence**. Choose **Incident Response** and give the approved public source. For this exercise the **Signal** is **Successful Exploitation**. You can include a brief approved summary in **Details** as JSON. Click **Add Evidence**.

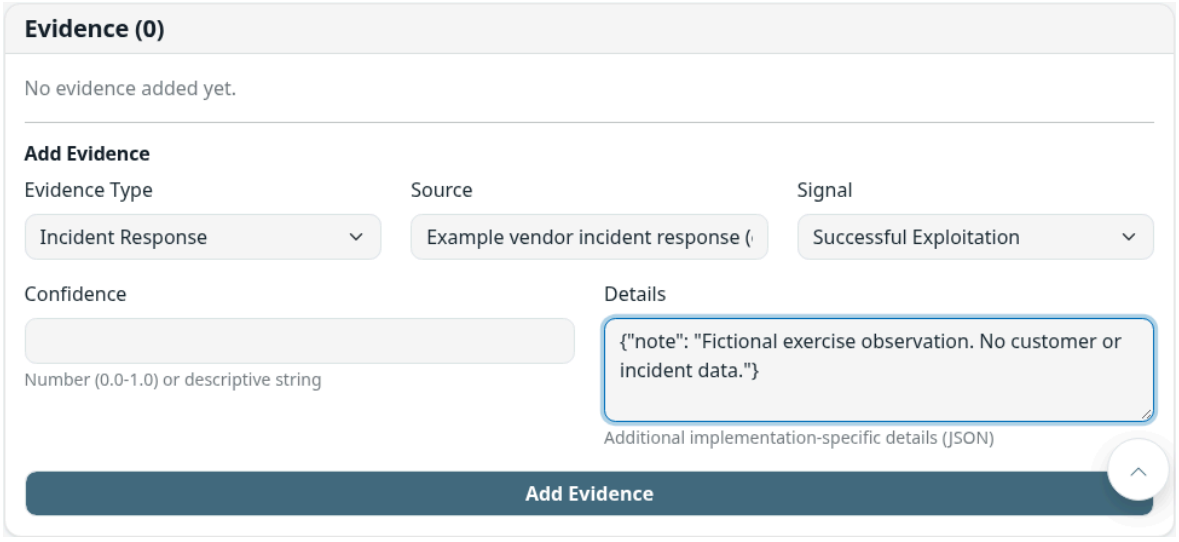


Figure 9.5: Attach an incident response evidence item to the saved entry.

4. Open the catalogue entry while signed out. Verify that it shows the correct ID, catalogue, status and scope. Find the evidence and confirm the source and signal agree with the approved summary.

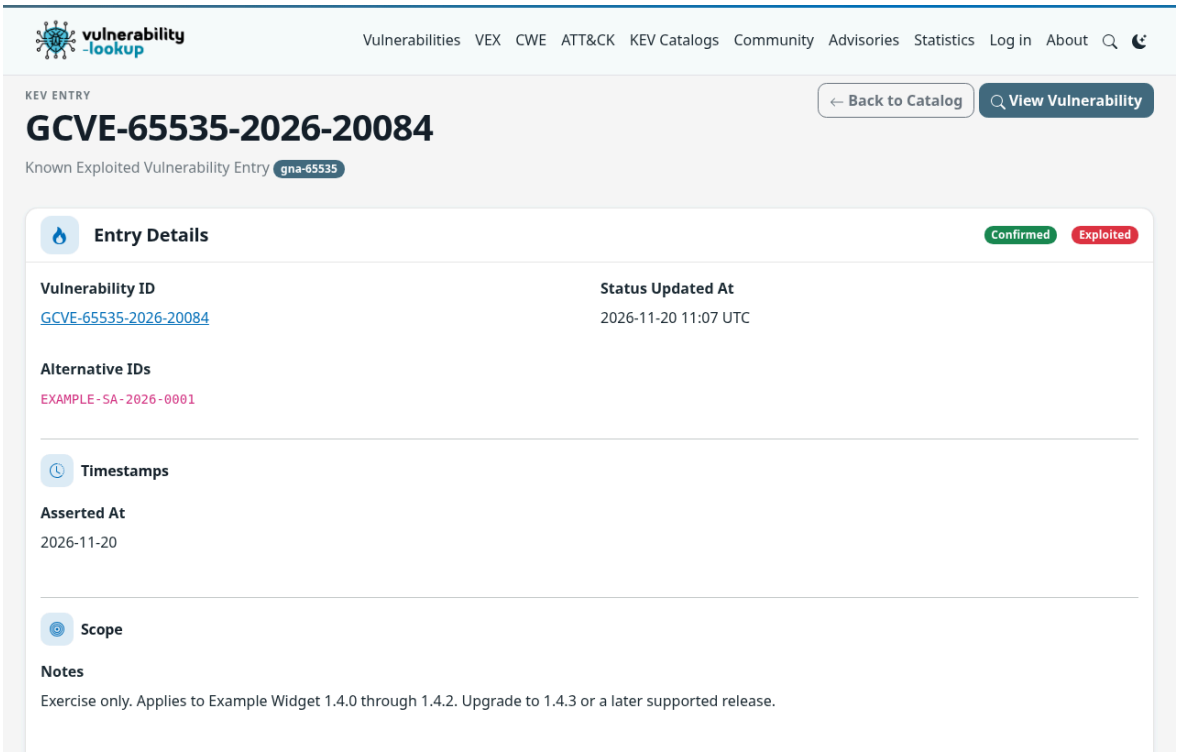


Figure 9.6: The signed-out vendor entry shows its local catalogue and confirmed status.

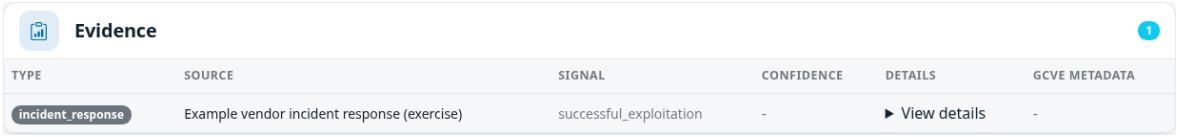


Figure 9.7: The vendor evidence item appears on the public entry.

9.3 List exploitation of a third-party advisory

A CSIRT may receive its own incident reports, see activity in telemetry, or receive information through an authorised CRA channel. Confirm which public vulnerability and product the evidence concerns. Check the relevant version or configuration and coordinate with the publisher where needed. A CSIRT can make its own scoped assertion about that vulnerability without editing or republishing the third-party advisory.

This walkthrough uses `CVE-2021-44228` for Apache Log4j2 and relies on CISA's public KEV listing. The CISA listing date, 10 December 2021, is a date of publication in that catalogue. The date of first exploitation is left unset in this example. CVE record · CISA public catalogue data

1. Look up the third-party vulnerability and verify the publisher, identifier, and affected product. If your instance collects the relevant vulnerability source, check the imported record and its references.

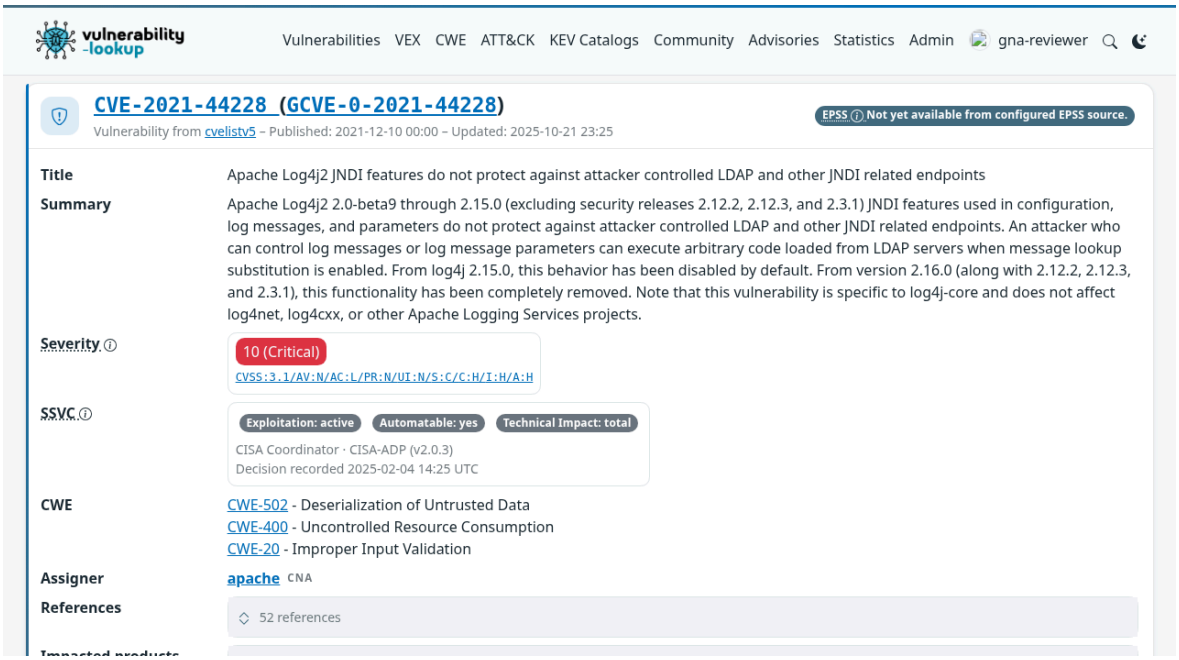


Figure 9.8: Check the imported CVE before creating a separate CSIRT assertion.

2. Go to **Admin → KEV catalog** and select **Add KEV Entry**. Paste the verified CVE into **Vulnerability ID**. The screenshot adds the equivalent GNA 0 identifier `GCVE-0-2021-44228` on a separate line in **Alternative IDs**. For the public CISA listing, use **Exploited** and **Confirmed**. State the scope and refer users to the upstream guidance, then **Save**. GNA 0 registry entry

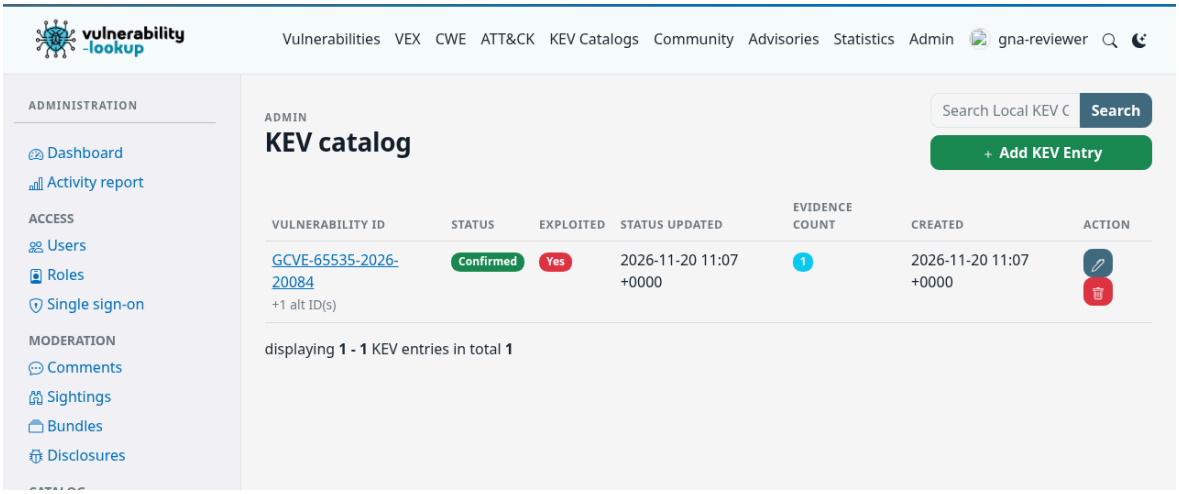


Figure 9.9: The administrator opens Add KEV Entry from the local catalogue.

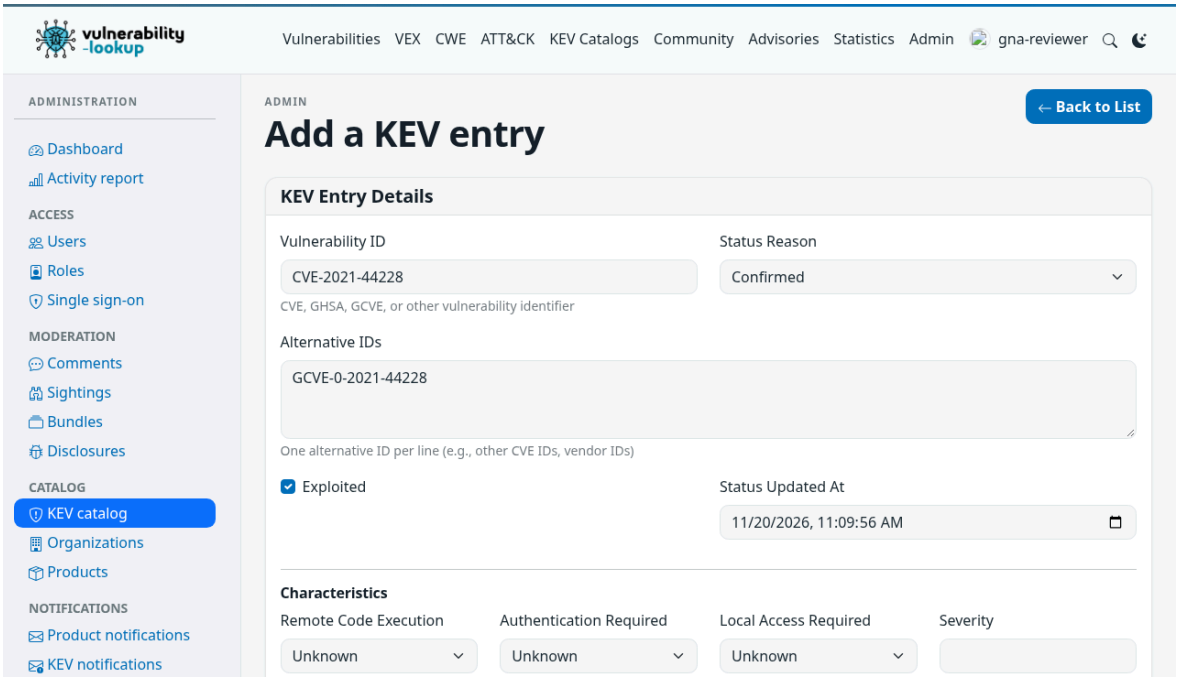


Figure 9.10: Use the imported CVE as the primary ID of the CSIRT's local entry.

3. In **Add Evidence**, select **Public Report** and name CISA's catalogue in **Source**. In **Details**, include the public URL and identify the date as the CISA listing date. If a public source gives no finer description of the observed activity, leave **Signal** unspecified. For direct CSIRT findings, choose the appropriate evidence type and publish only the summary and dates cleared for release.

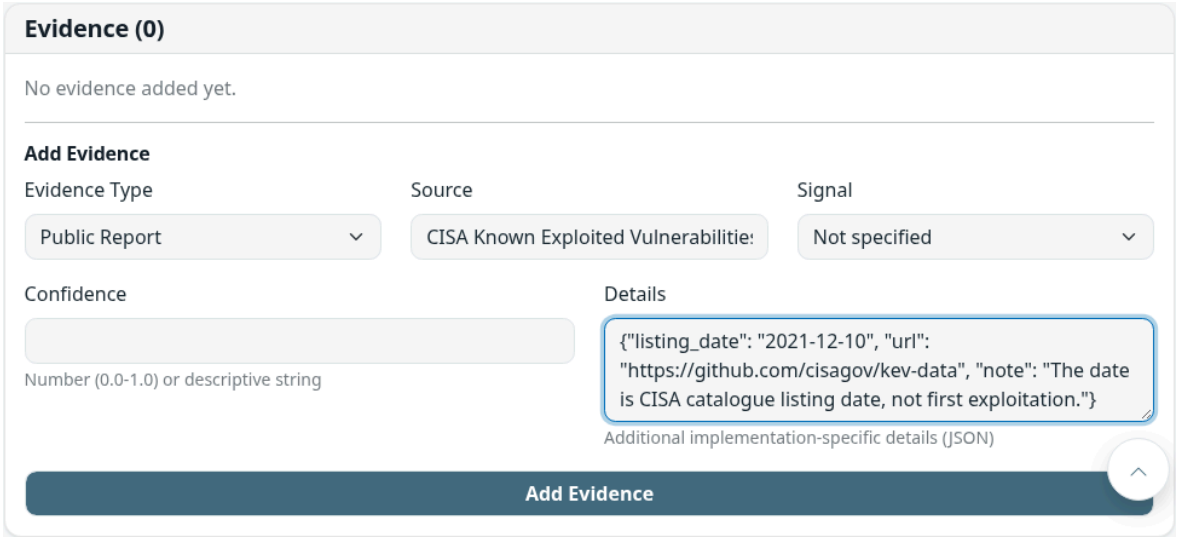


Figure 9.11: Record where the third-party exploitation information came from.

4. Open the entry while signed out. Follow **View Vulnerability** to check that it still opens the original CVE. Check that the entry is attributed to your catalogue and that CISA is named as the evidence source. Make sure readers can tell which product and versions the evidence covers, especially when a component is bundled in several downstream products.

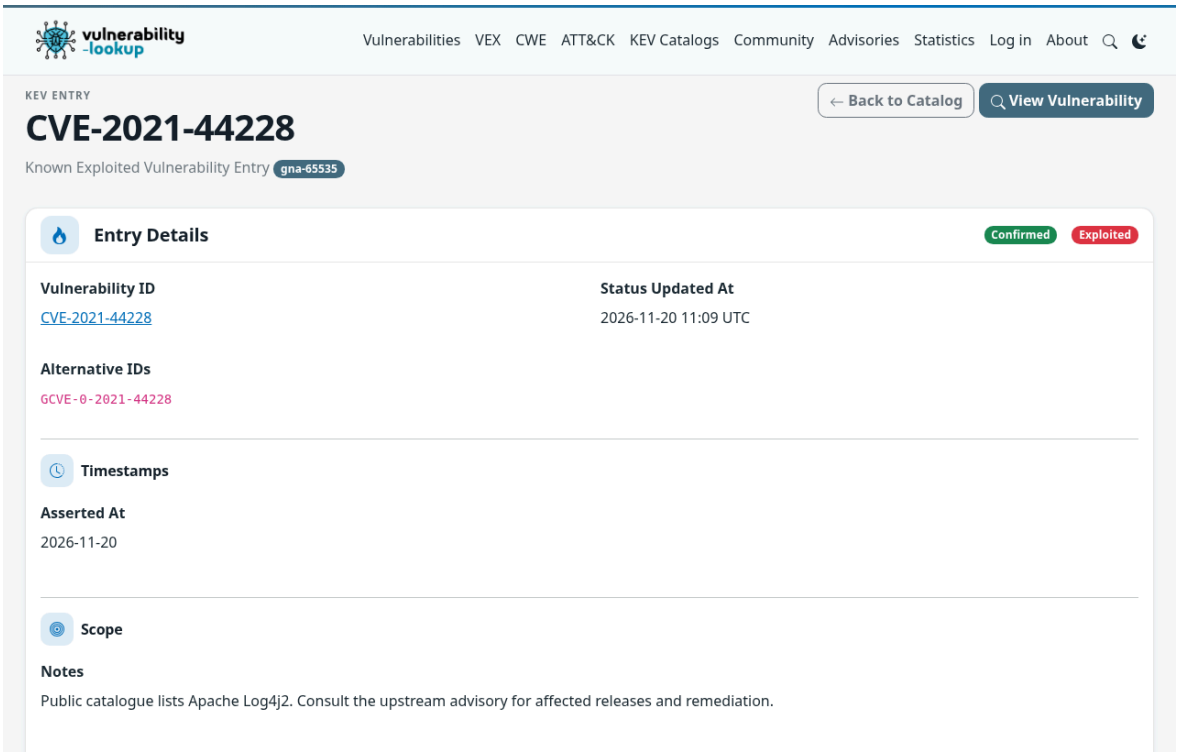


Figure 9.12: The CSIRT's entry references the existing CVE and offers a link back to it.

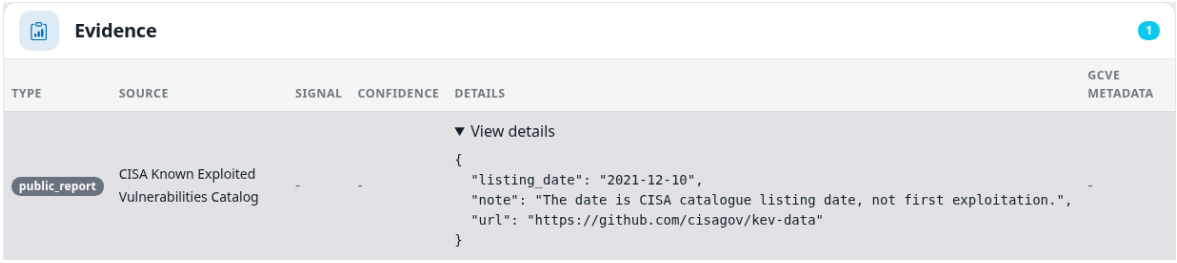


Figure 9.13: Readers can expand the public evidence to see the source and its listing date.

9.4 Verify and correct assertions

1. **Check the catalogue while signed out.** Open `/known-exploited-vulnerabilities-catalog/` and confirm both entries are present. Follow their detail links and check that each points to the right vulnerability and contains only approved evidence.

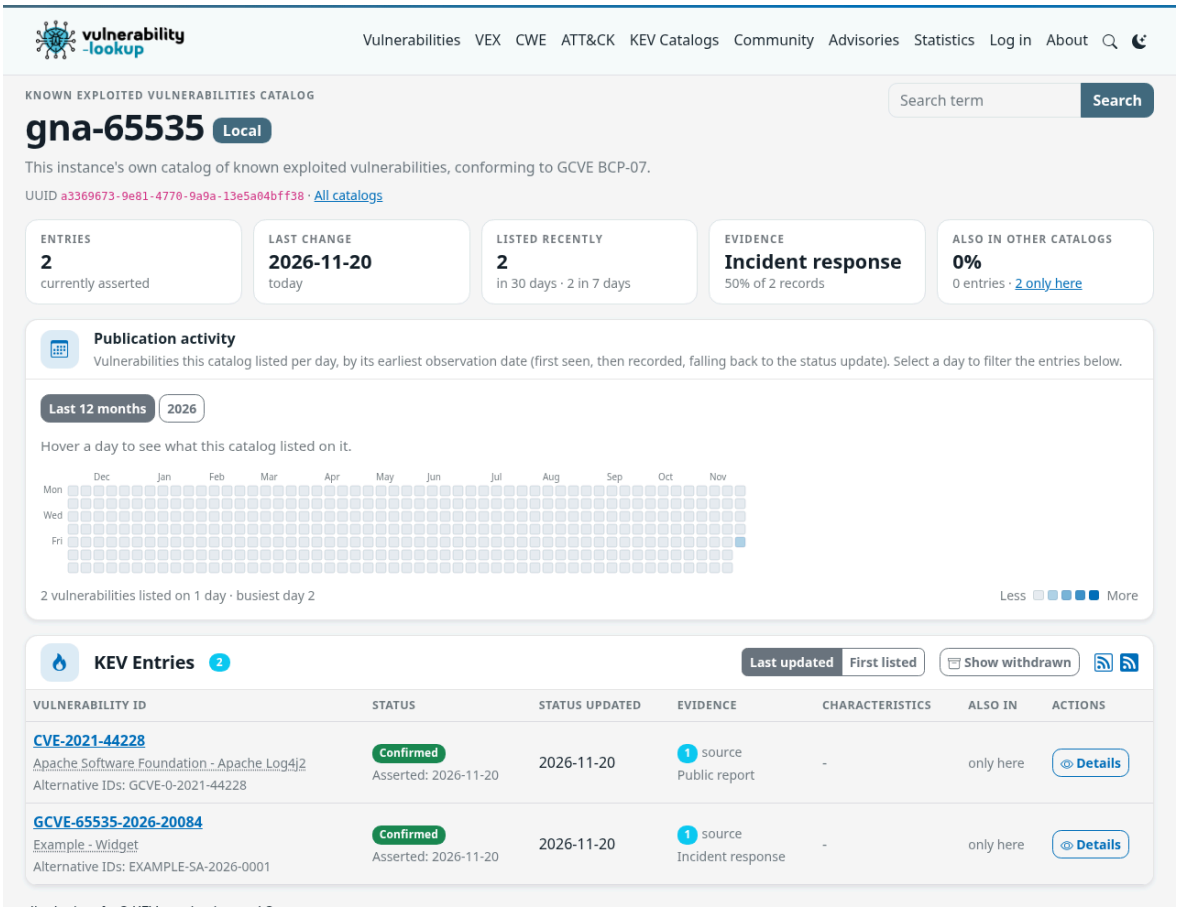


Figure 9.14: One local catalogue contains assertions about the vendor's own advisory and the third-party CVE.

2. **Verify the machine-readable version.** Find the matching `vulnerability.vulnId` in the BCP-07 export. Compare the entry UUID, origin, status, and evidence with the public page. An exploited sighting created from the KEV entry is derived from the same assertion. Count the underlying evidence only once.
3. **Correct the existing entry.** Use **Edit Entry** to update evidence or narrow the scope, record when the decision changed, and inform downstream consumers if their interpretation needs to change. If your organisation retracts its local assertion, open it and choose **Withdraw**. The current list drops the entry and its derived exploited sighting is removed. **Show withdrawn** and the NDJSON export let consumers see the retained entry and evidence under the original UUID. Check that the export now shows `status.exploited` as `false` and `status.status_reason` as `withdrawn`. BCP-07 withdrawal and reassertion

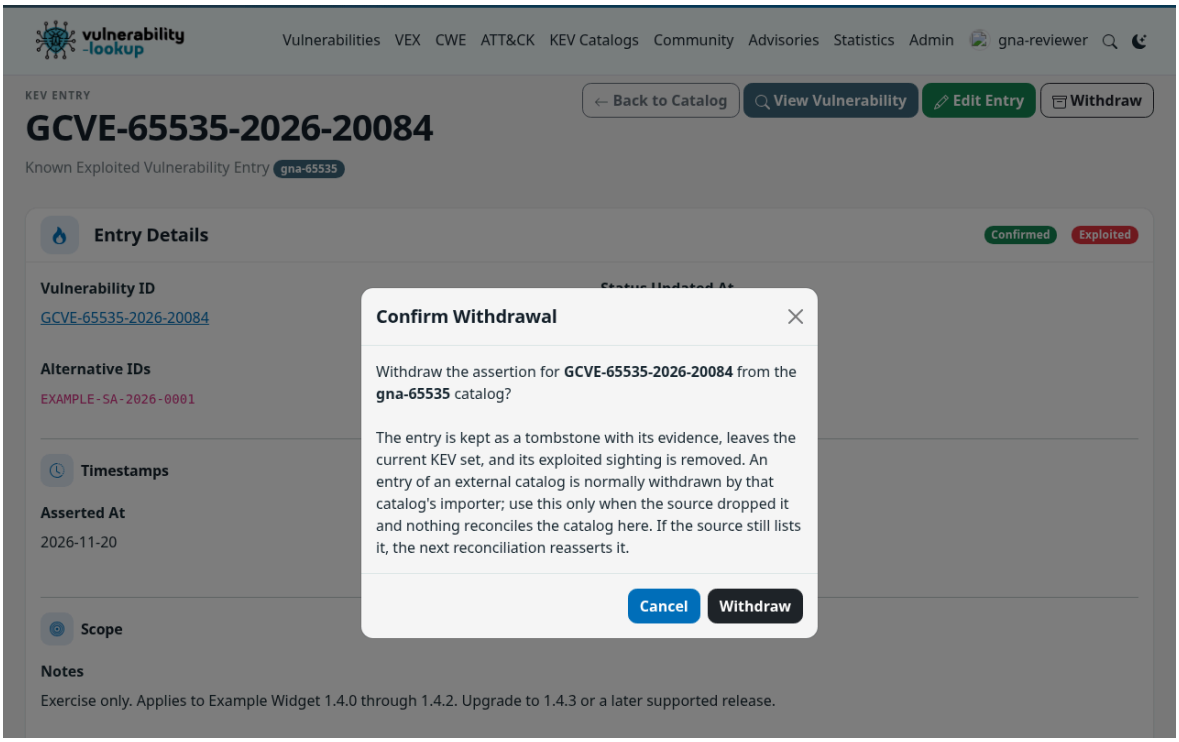


Figure 9.15: Confirm that you are withdrawing the selected entry from your own catalogue.

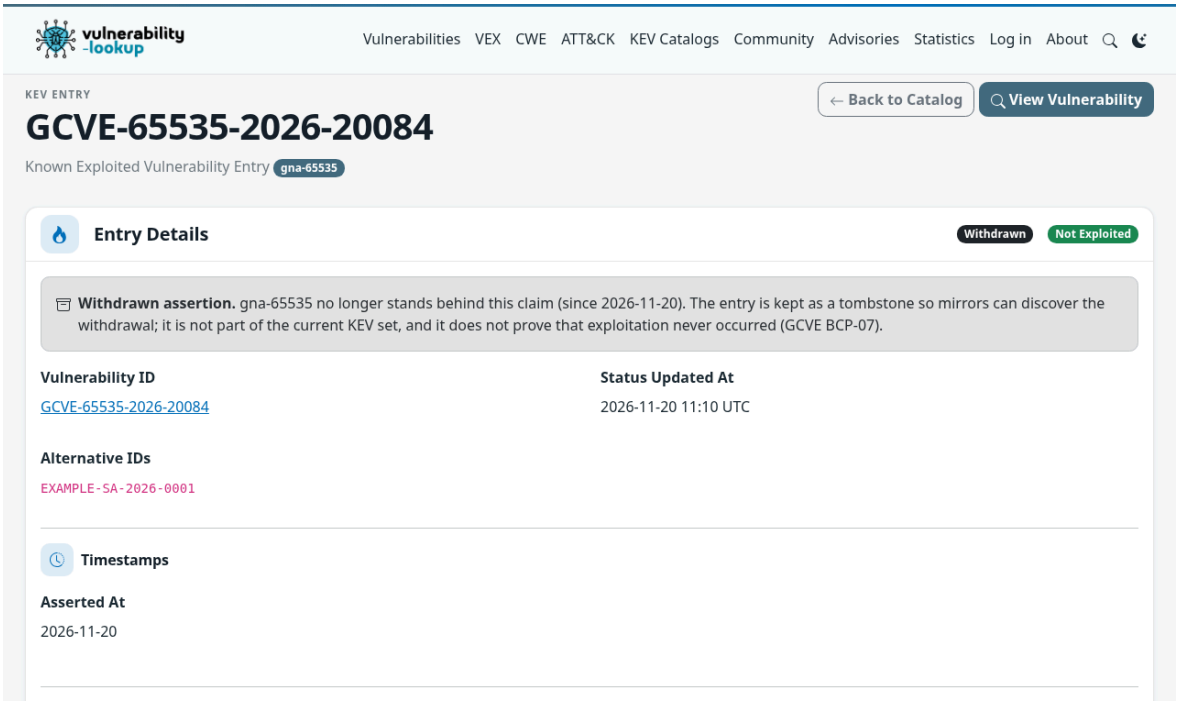


Figure 9.16: The signed-out detail page preserves a withdrawn assertion for reconciliation.

9.5 Share and synchronise BCP-07 assertions

Vulnerability-Lookup publishes KEV entries in the GCVE BCP-07 assertion format. The catalogue export uses newline-delimited JSON (NDJSON) with one assertion per line. This is the Example Widget assertion from the exercise, pretty-printed with some optional fields omitted. GCVE BCP-07 KEV assertion schema

```
{
  "uuid": "112389b3-5b3e-4288-a2c1-8ebca9beacc1",
  "vulnerability": {
    "vulnId": "GCVE-65535-2026-20084",
```

```
    "altId": ["EXAMPLE-SA-2026-0001"]
  },
  "status": {
    "exploited": true,
    "status_reason": "confirmed",
    "status_updated_at": "2026-11-20T11:07:19+00:00"
  },
  "gcve": {
    "origin_uuid": "a3369673-9e81-4770-9a9a-13e5a04bff38",
    "object_uuid": "112389b3-5b3e-4288-a2c1-8ebca9beacc1",
    "gna": 65535
  },
  "evidence": [
    {
      "source": "Example vendor incident response (exercise)",
      "type": "incident_response",
      "signal": "successful_exploitation"
    }
  ]
}
```


The entry's `uuid` identifies the assertion, and `gcve.origin_uuid` identifies the instance that published it. `vulnerability.vulnId` identifies the vulnerability. Several catalogues can publish their own assertions about that same vulnerability and cite different evidence. For a third-party entry, the origin still identifies the catalogue that made the assertion, while `evidence[].source` can credit a source such as CISA. The export can also carry observation times and scope notes.

9.5.1 Share a catalogue with another instance

1. **On the publishing instance,** share its public HTTPS base URL and the `local_instance_uuid` returned by `/api/system/configInfo`. Give file consumers `/known-exploited-vulnerabilities-catalog/export.ndjson` for the full catalogue, including withdrawn entries. Subscribers can use `/known-exploited-vulnerabilities-catalog/feed.atom` for recent changes. The built-in synchronisation service pulls BCP-07 assertions through the publishing instance's public `/api/kev/` endpoint. KEV feeds · Synchronisation service
2. **On the receiving instance,** go to **Admin → Remote instances → Add a remote instance**. Enter the publisher's **Instance UUID**, a name, and its base URL in **Address**. Enable **Active** and **Sync KEV Entries**. Keep **Sync local objects only** enabled to pull assertions created by that publisher. Clear the other sync toggles if you only need the KEV catalogue, then select **Save**.

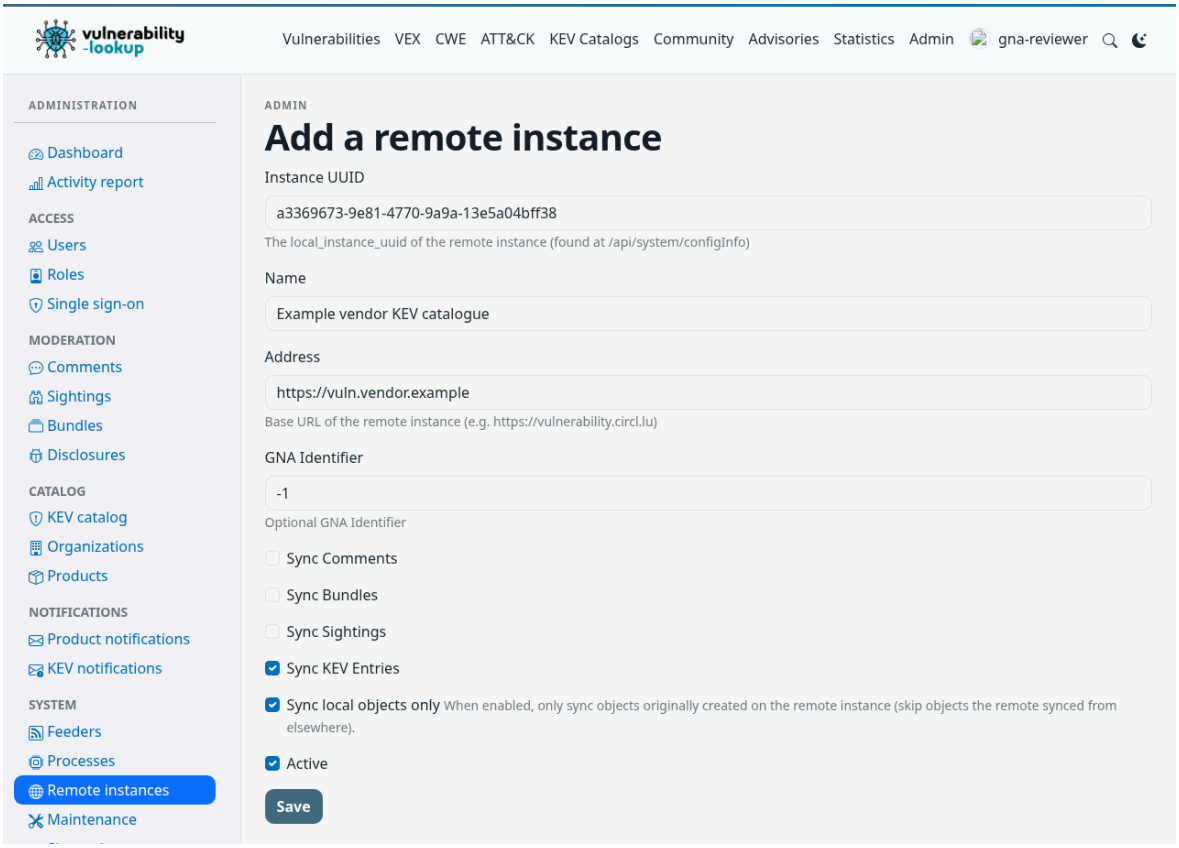


Figure 9.17: Configure the receiving instance to pull the publisher's own KEV assertions.

3. **Run and verify the pull.** In **Admin → Maintenance**, use **Trigger Sync from Remote Instances** with an existing account login. Run the sync service on a schedule to keep the receiving instance current. After a pull, check **Last sync** on the remote instance settings, and inspect the receiver's export at `/known-exploited-vulnerabilities-catalog/export.ndjson?catalog_uuid=<publisher-uuid>`. Replace `<publisher-uuid>` with the publisher's instance UUID. Check that the entry retains the original UUID and origin, with matching status and evidence. Synchronisation service
4. **Verify corrections in the receiving export.** After changing the evidence on a shared entry, save the assertion in **Edit Entry** so the publisher advances its status timestamp. In the version shown, incremental pulls filter by entry creation date. If an older correction or withdrawal is missing, use **Reset last sync** on the receiver's remote instance settings and run the pull again. Verify that a withdrawal updates the existing UUID to `withdrawn` with `exploited` set to `false`.

9.6 Use exploitation evidence in CRA reporting

The EU Cyber Resilience Act (CRA) defines an actively exploited vulnerability by reliable evidence that a malicious actor exploited it in a system without permission of the system owner. For a manufacturer, a KEV entry can help document the exploitation assessment and point users to remediation after public release is cleared. If you learn about exploitation through a third-party component or catalogue, assess whether the affected component and conditions apply to your own product. In the restricted case, record the evidence and its provenance, the affected product and versions, when your organisation became aware, and what protective measures are available. Use that history to prepare and update the notification. ENISA Single Reporting Platform FAQ

For manufacturers in scope, Article 14 reporting for actively exploited vulnerabilities applies from 11 September 2026. Submit the early warning without undue delay and within 24 hours of awareness, and the vulnerability notification without undue delay and within 72 hours of awareness. Submit the final report no later than 14 days after a corrective or mitigating measure becomes available. Use the CRA Single Reporting Platform (SRP) for these reports. Start reporting on the statutory timeline while the public release is being coordinated. CRA Articles 14 and 16 · ENISA reporting guidance · BCP-02 Appendix A

If your CSIRT receives a CRA notification, use the authorised channels for triage and coordination. Verify the product and vulnerability match, record the provenance and applicable restrictions internally, and coordinate any public wording and timing with the responsible parties. When release is cleared, the local catalogue can carry a scoped statement and approved evidence. Keep the notification, report identifiers, detailed observations, and identifying data in the restricted case. ENISA guidance on notification handling

PART 04

Use Cases

Coordinating disclosure across upstream and downstream teams.

CHAPTER 10

Coordinate an upstream and downstream release

Use this fictional tabletop to rehearse a single report affecting a shared library and a product that ships it. The CSIRT coordinates in a staging Vulnerability-Lookup instance. The library maintainer works upstream, a distributor ships the downstream product, and the distributor's advisory team publishes guidance for its users.

Stage	Action in Vulnerability-Lookup	Handoff to verify
Intake	Submit a fictional report about the Example Library and acknowledge it in **Discussion**.	The reporter receives the case reference and next update date.
Scope	Record confirmed upstream versions and separately track the distributor's shipped versions in **Notes**. Link the coordination ticket through **External reference**.	Both teams confirm the scope they own, including downstream version numbers that differ from upstream.
Coordination	Set the agreed disclosure date in **Review**. Record the release window, audience for limited sharing, and decision owner.	Upstream and downstream each confirm mitigation, fix delivery, and publication timing.
Publication	Rehearse the applicable CNA or GNA path. For an external identifier owner, rehearse the handoff using a fictional identifier recorded in the tabletop notes.	One owner confirms the identifier and both advisories use it for the shared issue.
Follow-up	Complete the staged disclosure and review the public content while signed out.	The case links to its public record and the advisories give product-specific update instructions.

Run the exercise again with the downstream update delayed. Have the teams decide whether to move the release or publish an earlier warning with available mitigations. Record who made the decision, how the reporter was informed, and the time of the next review. A complete rehearsal leaves each team with the same release plan and an owner for any follow-up.

BCP-02 · Appendix B, Multi-Party Coordinated Vulnerability Disclosure in Open-Source Ecosystems

CHAPTER 11

Draft a security advisory from a patch

When a security fix is available, a vendor or CSIRT can use **Patch2Vuln** in Vulnerability-Lookup's Vulniverse editor to draft a vulnerability record. Use the reviewed record to write the public advisory and give affected users a clear action. A vendor can confirm which releases ship the fix. A coordinating CSIRT can ask the maintainer to confirm scope, remediation, and what can be published. AI-assisted vulnerability records · Patch2Vuln

The screenshots show a fictional Example Desk patch in a local Vulnerability-Lookup 7.1.0 development instance. The upload and review run through the Patch2Vuln service with a fixed example model response. The GitHub URL illustrates the input format. Vulnerability-Lookup revision b4c82c1

11.1 Prepare the editor and select the fix

In `config/website.py`, enable `vulnerability_editor` in `WEB_MODULES` and set `VULNERABILITY_EDITOR = "vulniverse"`. Run a reachable Patch2Vuln service with a worker and an Ollama model. Enable `patch2vuln_enabled` and set `patch2vuln_url` in `config/patch2vuln.json`, then enable `panels.patch2vuln` in `config/vulniverse.json` and restart the website. The service credential can be set in the server's `PATCH2VULN_API_TOKEN` environment variable. Sign in with a publication operator account that can open the editor. Configure Patch2Vuln · Patch2Vuln service and worker

Confirm which commit or patch contains the security fix. Gather the maintainer's affected-release range, fixed releases on supported branches, and any interim mitigation. For an embargoed or internal patch, use a service and model approved to handle that source. The service holds the request while the job is queued or running and retains the result until deletion or expiry. Check its storage and access settings before uploading private material. Keep credentials, unrelated changes, and customer data outside the input. Patch2Vuln job data and retention

11.2 Submit a public commit or a patch file

Open **Advisories → New vulnerability advisory** and select **Patch2Vuln**. Choose the input that fits how the fix can be shared.

11.2.1 Public GitHub commit

Paste the URL of the specific fix commit under **GitHub commit URLs**, one per line. Patch2Vuln fetches the commit patch from GitHub on the server. Use a file upload for a repository hosted elsewhere. Group patches only when they fix the same vulnerability.

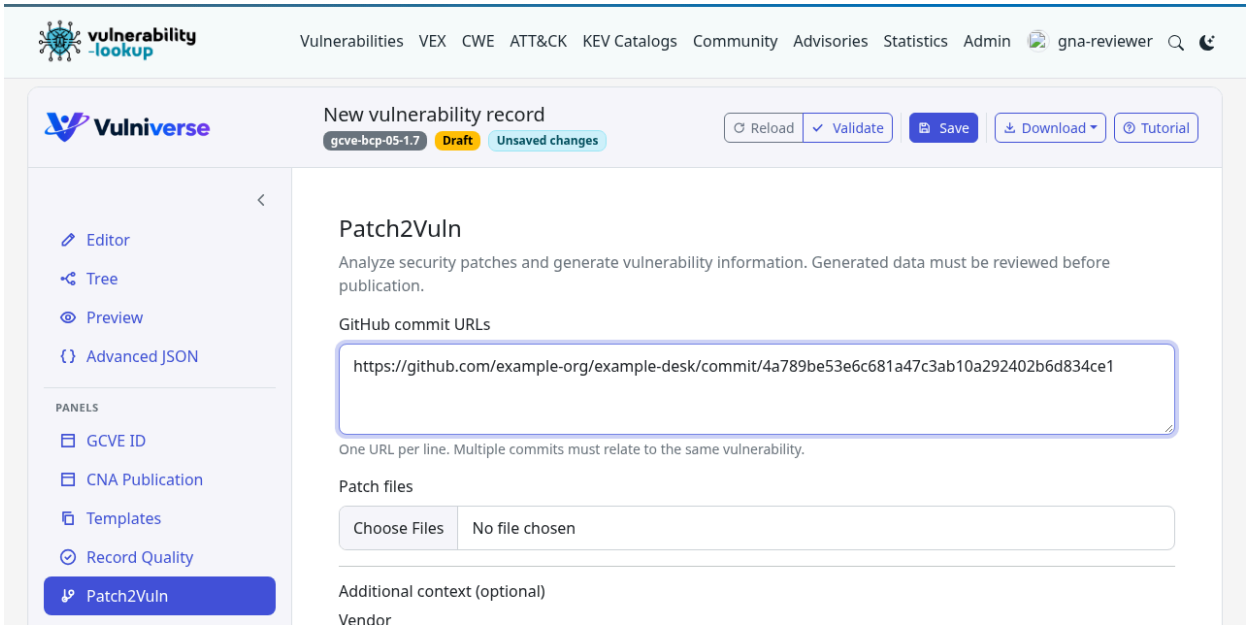


Figure 11.1: Paste the public fix commit into the GitHub commit input.

11.2.2 Internal repository or an existing patch file

Export the approved fix commit from a local checkout, or obtain the approved `.patch` or `.diff` from the maintainer. A Git patch may also contain the commit message, author, and internal filenames. Review that material against the scope your vendor or CSIRT is authorised to share.

```
git format-patch -1 --stdout "$fix_commit" > approved-fix.patch
```

Set `fix_commit` to the commit that contains the fix. Choose the resulting file in **Patch files**. The file is sent to the configured service, so the service can process it without direct access to the private repository.

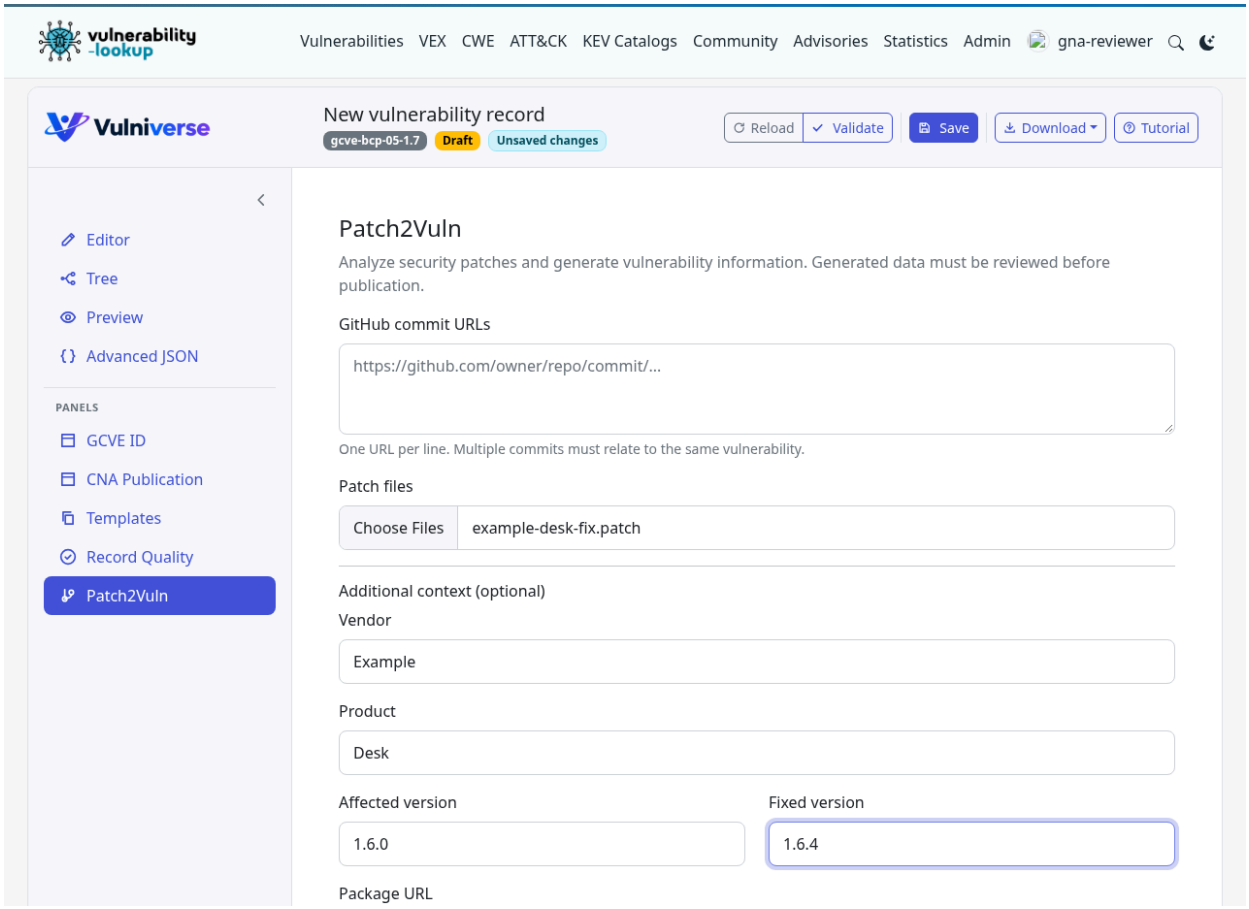


Figure 11.2: Upload the internal fix and supply the product and known version boundaries.

For either input, fill **Vendor**, **Product**, **Affected version**, and **Fixed version** with facts confirmed by the maintainer. Leave unknown context blank and resolve it during review. Use the correct **Version type** and include an unversioned **Package URL** when known. Select **Analyze patches**. In this release, the panel accepts up to five inputs and 4 MiB of uploaded patches per job. Wait for **Analysis completed successfully** before inspecting the result. Patch2Vuln inputs and limits

11.3 Select, verify, and edit the result

1. Expand **View generated record** and **View analysis details**. Compare the proposed title, description, versions, and solution with the fix and the maintainer's release information. Review any assumptions and resolve differences with the vendor or upstream project.
2. Select the fields you want to use and click **Apply selected fields**. The selected fields replace their current values in the editor. Record identifiers and publication metadata are preserved. In the example, the operator starts with the title, descriptions, affected products, and solution. Keep other fields unchecked until you have independently verified them. Carry over the AI provenance

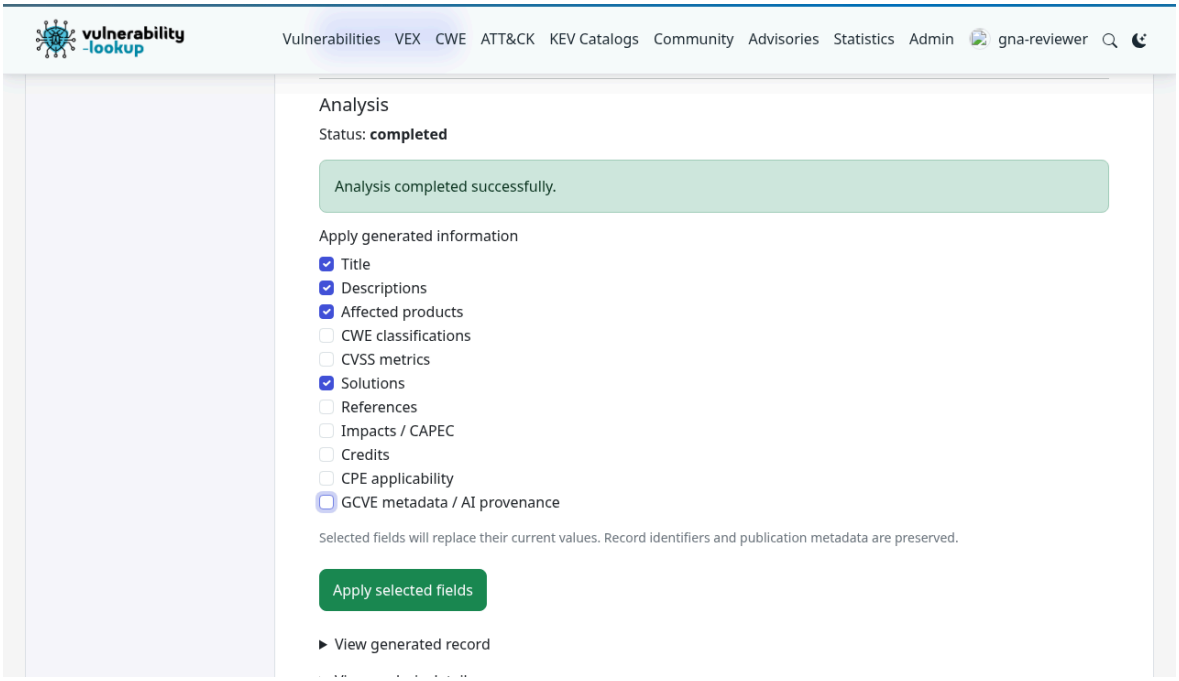


Figure 11.3: Apply just the fields that are ready for review in the editor.

3. Review the applied fields in **Editor**. Confirm the impact and affected release ranges, add update instructions and any workaround, and use approved public references. Check that references, credits, comments, and provenance contain only material cleared for release. Confirm severity and any exploitation claims using your team's evidence.

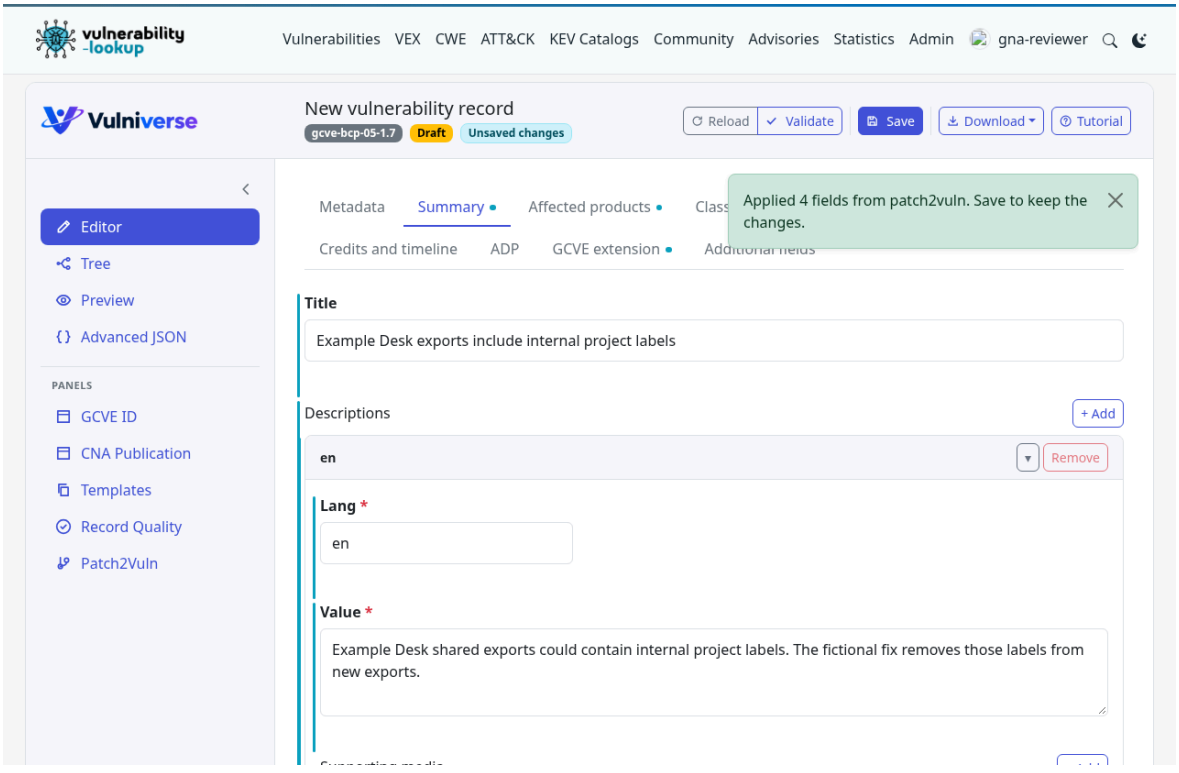


Figure 11.4: The selected draft fields are now editable in Vulniverse.

4. Open **Preview** and use **Validate** to check the record format. If enabled, use **Record Quality** to identify missing fields. Have the release approver verify the facts and public wording before publication. Use **Download** to hand off a draft in JSON, HTML, or PDF if the action is enabled. Keep the draft in your restricted case or release system. Write and evaluate a vulnerability record · Vulnerability editor

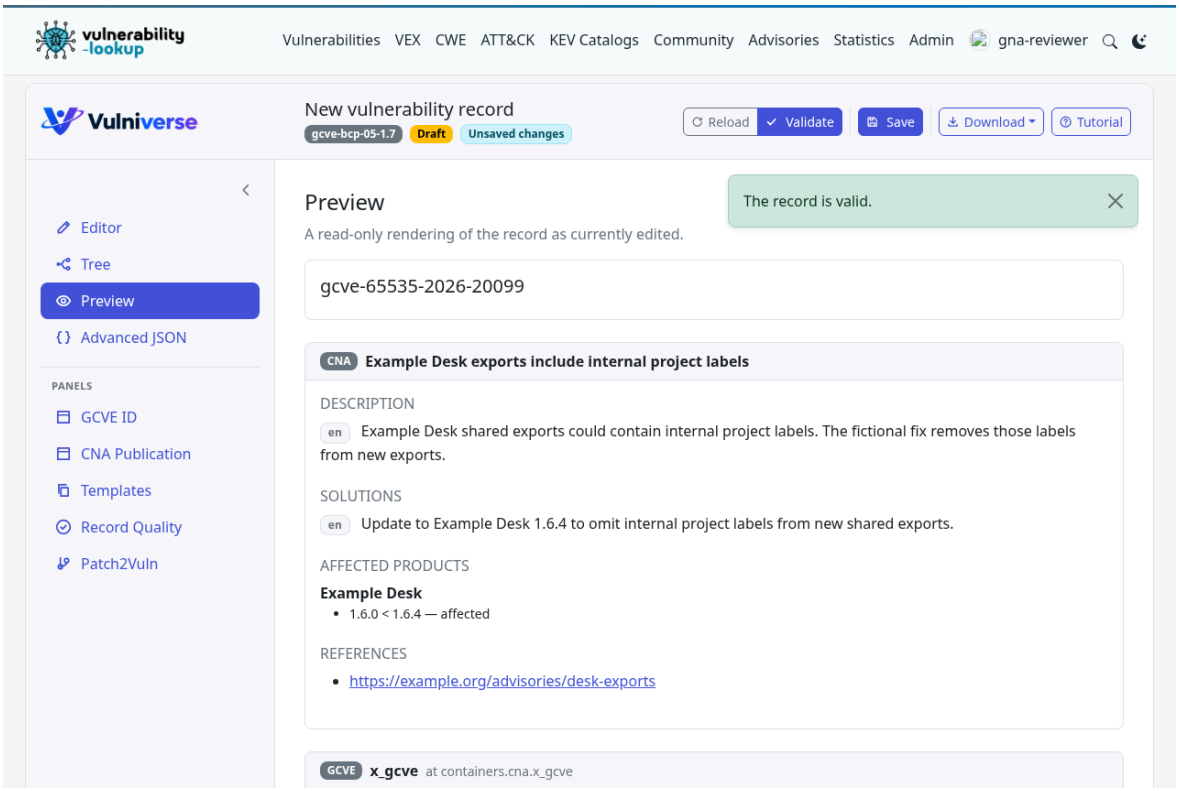


Figure 11.5: Check what the draft will show before moving to publication.

11.4 Record AI assistance and patch provenance

By default, Patch2Vuln adds the following GCVE provenance extensions to its generated CVE-format draft. Expand **View generated record** and find them under `containers.cna.x_gcve[].extensions`.

Extension	What it records
BCP-05-X-01	The level and scope of AI assistance, the model name or identifier and source, and the human review status. A newly generated draft starts with `ai_level: generated` and `review_status: none`.
BCP-05-X-02	How the patch analysis was run and what it produced, including the generator, timestamp, model, patch source and SHA-256, input truncation, confidence, assumptions, and assessment rationales.

When applying the draft to a CVE or GCVE record, select **GCVE metadata / AI provenance** with the fields you want to carry over, then click **Apply selected fields**. Inspect both extensions in **GCVE extension** or **Advanced JSON**. Keep exact inference settings, such as temperature, in your restricted run log when you need to reproduce the analysis. Patch2Vuln generation provenance · Vulnerability-Lookup AI-assisted records

Before release, review the extensions for private repository paths, commit subjects, and other details that need clearance. Set `review_status` to reflect the work actually completed. Remove the X-02 `draft` flag once publication review is complete. Check that both extensions remain in the exported record.

The extensions give publishers machine-readable evidence of AI involvement and human review, which can support transparency and documentation under the EU AI Act. Assess the applicable duties for your role and how you publish the advisory. Article 50(4) addresses AI-generated or manipulated text published to inform the public on matters of public interest. It provides an exception where the content has undergone human review or editorial control and a natural or legal person holds editorial responsibility for the publication. Record that responsibility and provide any required disclosure to readers. EU AI Act Article 50

11.5 Write and release the advisory

Use the approved description and remediation to write a concise advisory for affected users. The Example Desk exercise would tell users that shared exports from versions 1.6.0 through 1.6.3 could include internal project labels and that new exports produced by 1.6.4 omit them. The advisory should tell users where to obtain 1.6.4, what to do about earlier exports if needed, and where future corrections will appear.

Choose the identifier and publication route agreed with the coordinator. Share the reviewed draft with the assigned publisher if another team owns that route. At release, publish the record, link it and the user-facing advisory, then verify both while signed out. If there is an associated Vulnerability-Lookup disclosure, add the public identifier in **Review** and set the case to **Disclosed** at the agreed time. Local and CNA publication · GNA publication · Advisory and follow-up

Appendices

APPENDIX A

Case and release checklists

A.1 Before approval

- The case has an owner, a backup, a release approver, and a dated next action.
- The reporter has received an update in **Discussion** and confirmed their credit preference. Any private case references and vendor coordination notes are in **External reference** or **Notes**.
- The maintainer has confirmed affected versions, mitigations, and fix availability. Any gaps are documented for the release decision.
- Participating teams have the release time and time zone, identifier owner, and advisory owner. The date in **Review** reflects that plan.
- The public draft has been reviewed for sensitive case data, including reporter details and internal references.

A.2 At release

- Publish the vulnerability record through the agreed channel and confirm the public identifier.
- Publish the advisory with version-specific guidance and verify that the update or mitigation instructions are reachable.
- Add the identifier in **Review → Vulnerability ID**, select **Disclosed**, and save at the agreed time.
- Verify the public pages and configured distribution channels while signed out. Send links to the reporter and participating teams.

A.3 After release

- Assign follow-up ownership and propagate corrections to the advisory and vulnerability record.
- Review time to acknowledgement, time to publication, and any missed handoffs. Record what the team will change before the next case.

See Investigation and coordination and Publish the advisory and follow up for the corresponding workflows.

APPENDIX B

References

B.1 Map from GCVE BCP-02 to this book

The GCVE Working Group's Practical Guide to Vulnerability Handling and Disclosure (BCP-02, v2.0) is the source for the handling lifecycle. It is published under CC BY 4.0.

BCP-02 section	Where to apply it
Roles and Responsibilities, Preparing for Vulnerability Handling	Roles and handoffs, Prepare your instance
Receiving and Handling Vulnerability Reports	Intake and triage
Investigating and Resolving Vulnerabilities, Communicating with Submitters and Users, Coordinated Vulnerability Disclosure	Investigation and coordination
Publishing Advisories, Best Practices	Publish as a CNA, Publish as a GNA, Publish the advisory and follow up
Appendix A, EU Cyber Resilience Act Reporting	Decide when to disclose and Use exploitation evidence in CRA reporting. See BCP-02 Appendix A for the reporting process.
Appendix B, Multi-Party Coordinated Vulnerability Disclosure in Open-Source Ecosystems	Coordinate an upstream and downstream release

B.2 Product documentation

- Vulnerability-Lookup coordinated vulnerability disclosure
- CNA Publication Service
- Minimal CNA/GNA Publication Instance
- Vulnerability Editor
- Roles and Permissions

B.3 Publication standard and layout reference

- GCVE BCP-03, Decentralized Publication Standard
- GCVE BCP-07, Known Exploited Vulnerability Assertion Format
- Chapter-per-directory structure and HTML and PDF styling adapted from the MISP Book. Styling credits and licenses