ONAP Vulnerability Management

ONAP Vulnerability Management and the onap-security mail alias are only to report issues against the ONAP software itself. It is NOT to be used for any issues related to tools and infrastructure (DNS, email, web, etc.)

- **If you would like to report a vulnerability against general project infrastructure (such as DNS, web or email services)**, please go to http://support.linuxfoundation.org/ Project Services Infrastructure Operations and file a bug.
- The ONAP Project does not pay bug bounties.

Glossary

Term	Definition
Embargo	A time period where key ONAP stakeholders have access to details concerning the security vulnerability, with an understanding not to publish these details or the fixes they have prepared. The embargo ends with a coordinated release date (CRD). (adapted from source)
Subject Matter Expert (SME)	A developer or other specialist who can provide contextual information that helps to determine the validity and impact of a potential security vulnerability.
Security SME	A security SME is a specialist who is familiar with the ONAP security vulnerability procedures and security in general.
Peer reviewed	In the context of a patch, the term peer reviewed refers to the patch having been reviewed by the ONAP vulnerability sub-committee and any other relevant key stakeholders. There is not yet a strict definition of the number of people who need to have reviewed the patch, or how they provide sign off.

Credits

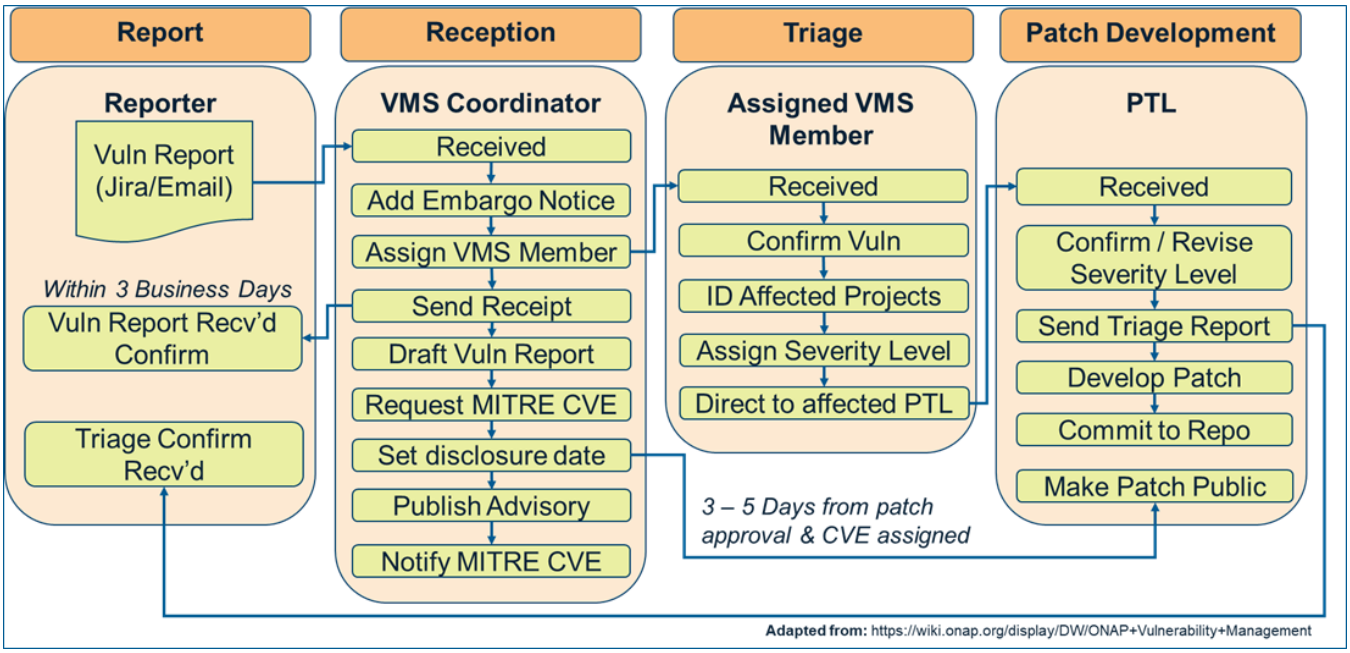
This document is strongly based on Vulnerability Management Process defined by Open Stack Community.

The whole document is licensed under Creative Commons Attribution 3.0 License.

Additionally, in an attempt to avoid re-inventing the wheel, the ONAP vulnerability management process borrows from the following procedures:

- The Linux Kernel process for reporting security issues
- The OpenDaylight vulnerability management process
- Recommendations for a minimal security response process
- The fd.io vulnerability management process

Vulnerability Management Process Overview



Vulnerability Management Process

The ONAP vulnerability management subcommittee (VMS) is responsible for coordinating the response to a reported vulnerability from initial reporting until coordinated disclosure.

Members of the team are independent and security-minded people who ensure that vulnerabilities are dealt with in a timely manner and that downstream stakeholders are notified in a coordinated and fair manner. Where a member of the team is employed by a downstream stakeholder, the member does not give their employer prior notice of any vulnerabilities. In order to reduce the disclosure of a vulnerability in the early stages, membership of this team is intentionally limited to a small number of people.

This activity is approved and supported by the ONAP TSC and operates under the ONAP vulnerability sub-committee. The sub-committee functions are as described below. The committee has a chair, appointed by the membership from among the membership, who is responsible for seeing that work proceeds and serves as a point of contact for the TSC and community to the vulnerability sub-committee. The chair and membership, as well as pointers to this charter and the relevant email lists are document at link-to-page.

Supported projects and versions

All ONAP projects are currently in scope for vulnerability support. The participants of the ONAP projects are expected to support the ONAP vulnerability procedures when required.

As ONAP is very young project with a lot of code coming in every release. Even through we are interested in receiving bugs for all ONAP releases that are currently in use, we will develop patches **ONLY FOR THE LATEST RELEASE** and **FOR THE MASTER BRANCH** (next version under development) . Unfortunately ensuring security in very early stages of the project is not always possible, that is why we declare three first releases (Amsterdam, Beijing, Casablanca) as **unsupported in terms of security bug fixes**. Dublin is going to be first version that will be supported as described by above rule.

Third party components

Third party components (i.e. dependencies) are only in scope for security support if they are statically compiled or otherwise bundled by an ONAP project. Dynamically linked dependencies should patch security issues independent of ONAP.

Dependencies on managed functions (eg. VNFs)

Vulnerabilities of managed functions (e.g. VNFs) are out of the scope of ONAP, however if an ONAP vulnerability has a dependence with a managed function, the managed functions vulnerability procedures will be used to coordinate the issue.

The Process

Notice

All the tasks mentioned below should be executed by VMS coordinator who is chosen and assigned by VMS based on their internal agreements on case by case basis.

Overview

Reception

A report can be received either as a ticket in Vulnerability Reporting Jira Project, email to onap-security@lists.onap.org or as a private encrypted email to one of the VMS members .

Steps that has to be completed depend on reception method:

Reception via Jira Ticket

1. Edit the ticket description by adding embargo notice in the beginning **(VMS team)**
2. Add "Uncategorised" label to the ticket **(VMS team)**
3. Confirm bug reception by assigning the task to one of VMS members and adding a reception confirmation comment. **(VMS team)**

Reception via email

1. Create new Jira ticket and post embargo notice and the content of original message re-encrypted with public keys of other VMS members. **(email recipient)**
2. Add "Uncategorised" label to the ticket **(email recipient)**
3. Assign the task to one of VMS members (capable of decrypting the bug content) **(email recipient)**
4. Send a PGP signed reception confirmation email (**DO NOT** include the original message in plain text). **(email recipient)**

VMS should do its best to provide a prompt confirmation to the reporter. Bug reception should be confirmed no later than **within 3 business days**.

It's worth noting that every new bug in the Vulnerability Reporting Jira Project by default is visible only to reporter and VMS members.

Triage

The bug must then be confirmed to be a security problem and assigned initial severity level. This may require the inclusion of additional subject matter experts to determine if the problem needs to be treated as a security flaw. If the bug is determined not be a security issue then a statement should be added indicating the justification. The bug should then be opened and fixed by following the normal development process.

Steps to be completed

1. Identify affected projects and versions
2. Update bug description with data mentioned above
3. Perform initial severity assessment
4. Label task with a suitable severity level and add impact description
5. Add PTL or security responsible person of affected project to the bug

Extra steps for critical vulnerabilities

1. Extreme caution is necessary while handling this bug.
2. If the bug has been reported via encrypted email, no plain text communication should be used
3. Secure communication channel with PTL or project security contact point should be established (GPG, zoom with e2e encryption enabled etc)

All Vulnerabilities

1. PTL or project security contact point should receive bug details to confirm initial severity
2. Severity level should be fixed:
    a. If PTL or project security contact point agrees with initially assigned severity a "Severity-confirmed" label should be added to the task.
    b. If PTL or project security contact point disagrees with initially assignment, a clear justification should be provided, severity level updated and "Severity-confirmed" added to the task.
3. If a bug has been received via email the triage confirmation email should be sent to the reporter.

Hardening opportunities

1. If received ticket is not a security bug which can be fix with a patch of reasonable size but a good hardening idea, the ticket visibility should be set to public and "Hardening" label should be added
2. After public discussion, it will be determined if resources should be assigned for this task.

Non-Security bugs

1. If bug has been classified as a non-security, the ticket should be made publicly visible

2. PTL of impacted project is responsible for further handling of this bug

Patch development

The PTL or project security contact point is responsible for fixing the bug or delegating the work to the subject matter experts. Even through patch development can be delegated by PTL or project security contact, only the VMS has a right to add new people to the ticket. Thus, the PTL should explicitly request access for additional developers by adding a comment with their LFID.

A fix should be prepared against current master branch and other supported affected branches and attached to the ticket. **Do not sent it to the public code review system (gerrit) unless the ticket is already public.**

Security fixes, especially critical should be treated as **highest-priority tasks**. If project delays are encountered at this or any subsequent stage of the process, the VMS and other interested parties may escalate the issue to the TSC but without providing any details on bug itself apart from reporter, severity, impacted project and versions.

Steps to be completed

1. Develop the fix **(PTL or delegate)**
2. **Do not sent it to the public code review system (gerrit) unless the ticket is already public.**

Patch review

Once the patch has been attached to the ticket, patch should be reviewed and pre-approved by PTL or delegated committers. Privately-developed patches need to be pre-approved so that they can be fast-tracked through public code review later at disclosure time.

For public reports, usual public code review process apply.

Steps to be completed

1. Review the fix **(PTL and committers)**
2. Pre-approve fix by providing a comment to the ticket "Acked-by: Name Surname <email@domain.tld>" **(PTL, committers and VMS coordinator)**
3. When all required approval are collected, commit message should be updated and comments from the previous step should be added to the commit message **(VMS coordinator)**

Draft Vulnerability description

While the patch is being developed, the VMS coordinator prepares a vulnerability description that will be communicated to downstream stakeholders, and will serve as the basis for the Security Advisory that will be finally published.

The description should properly credit the reporter, specify affected versions (including unsupported ones) and accurately describe impact and mitigation mechanisms. The VMS coordinator should use the template below.

Steps to be completed

1. Prepare draft of vulnerability description

Review impact description

The description is validated by the reporter and the PTL.

Steps to be completed

1. Review the draft of vulnerability description **(PTL and commiters)**

Send CVE request

If reporter did not request for a CVE number on his or her own, VMS coordinator should attempt to obtain one to ensure full traceability. This is generally done as the patch gets nearer to final approval. The approved impact description is submitted through MITRE's CVE Request form. The *request type* is `Request a CVE ID`, the *e-mail address* should be that of the requester, and for critical reports the coordinator's OpenPGP key should be pasted into the field provided.

In the *required* section set the check boxes indicating the product is not CNA-covered and that no prior CVE ID has been assigned, select an appropriate *vulnerability type* (using `Other or Unknown` to enter a free form type if there is nothing relevant on the drop-down), set the *vendor* to `ONAP`, and the *product* and *version* fields to match the `affected project name` and `version` from the impact description. In the *optional* section set the radio button for *confirmed/acknowledged* to `Yes`, choose an appropriate *attack type* in the drop-down (often this is `Context-dependent` for our cases), check the relevant *impact* checkboxes, attempt to fill in the *affected components* and *attack vector* fields if possible, paste in the *suggested description* from the prose of the impact description (usually omitting the first sentence as it's redundant with other fields), put the `$CREDIT` details in the *discoverer/credits* field, and the bug URL (along with Gerrit URLs for patches if already public) in the *references* field. If the report is still private, note that in the *additional information* field like `This report is currently under embargo and no disclosure date has been scheduled at this time.`

At the bottom of the page, fill in the *security code* and click the *submit request* button. If some fields contain invalid data they will be highlighted red; correct these, update the *security code* and *submit request* again until you get a confirmation page.

Steps to be completed

1. Request CVE number from MITRE

Get assigned CVE

MITRE returns the assigned CVE. VMS coordinator adds it to the jira ticket, and retitles the bug to “$TITLE ($CVE)”.

Steps to be completed

1. Receive the assigned CVE number
2. Add received CVE number to the Jira ticket
3. Re-title the ticket to “$TITLE ($CVE)”

Embargoed disclosure

Once the patches are approved and the CVE is assigned, a signed email with the vulnerability description is sent to the downstream stakeholders by VMS coordinator or other designated VMS member. The disclosure date is set to 3-5 business days, excluding Monday/Friday and holiday periods, at 1400 UTC. No stakeholder is supposed to deploy public patches before disclosure date.

For non-embargoed, public vulnerabilities no separate downstream advance notification is sent.

Steps to be completed

1. Set up disclosure date with the reporter, PTL and required committers.
2. Add note about planned disclosure date to the ticket
3. Send a PGP signed email with vulnerability description to downstream stakeholders.

Coordinated disclosure

In preparation for this, make sure you have a PTL available to help pushing the fix at disclosure time.

On the disclosure hour, open ticket to public, push patches to Gerrit for review on master.

PTL and committers who pre-approved the patch should, as soon as possible add +2 on pushed patch and merge it.

Publish the ONAP Security Advisory and update the ticket title to “[OSA-$NUM] $TITLE”.

Embargo reminder can be removed at that point.

MITRE’s CVE Request form should be used again at this point, but instead select a *request type* of `Notify CVE about a publication` and fill in the coordinator’s *e-mail address*, provide a *link to the advisory* (the URL to official OSA), the *CVE IDs* covered, and the *date published*. Once more, fill in the *se curity code* at the bottom of the page and *submit request*.

Steps to be completed

1. Ensure that PTL and committers are available
2. On the disclosure hour:
    a. Remove embargo notice from ticket description
    b. Open the ticket to public
    c. Push attached patches for review on master
    d. Notify PTL and committers that patch is ready to be merged
3. Publish ONAP security advisory
4. Update the ticket title to “[OSA-$NUM] $TITLE”
5. Sent notification to MITRE about a publication

Handling public/leaked security issues

What is considered public?

- Any comment on a public forum, whether it be a mailing list, irc, twitter, or news group, that discloses the details of the flaw.
- Any commit or review comment that indicates that the change may be security related.

Public security issue workflow

There will be occasions where the vulnerability management process is not followed and the issue is publicly disclosed before reporting it to the vulnerability subcommittee. In this case it's important to properly identify the issue and create a task to make it traceable. As the flaw has been already disclosed there is no need to keep the Jira ticket private so it should be set to publicly available in a very beginning of the process. In general, standard vulnerability management process should be followed, just embargoed disclosure should be skipped.

Steps to be executed:

1. Create ticket with issue description in Vulnerability Reporting Jira Project **(VMS members)**
2. Make the ticket publicly visible **(VMS members)**
3. Assign the bug to one of VMS members
4. Perform bug triage and CVE request if necessary **(VMS coordinator)**
5. Send email containing triage results to ONAP TSC Chair and LFN representative (Kenny Paul and Jim Baker)
6. Rest of standard process should be followed, skipping embargoed disclosure step

Leaked security issue workflow

Occasionally it may happen that the bug reporter or some else break the embargo and disclose details of the flaw before official disclosure date. In this case a timely answer is extremely important.

Steps to be executed:

1. Make the related ticket publicly visible
2. If a patch has been already proposed push it immediately to gerrit
3. Skip embargoed disclosure.
4. Send email confirming that issue has been leaked to ONAP TSC Chair and LFN representative (Kenny Paul and Jim Baker)
5. Rest of standard process should be followed and finished as soon as possible.

References

1. Common Vulnerabilities and Exposure (https://cve.mitre.org/about/faqs.html)
2. CVE numbering authorities (https://cve.mitre.org/cve/cna.html)
3. CVE FAQ (https://cve.mitre.org/about/faqs.html#what_is_cve_identifier)