Proposal: Kubernetes bug bounty program

kaczorowski@google.com
Shared Feb 20 2018
THIS DOCUMENT IS SHARED PUBLICLY

Goals

To create a vulnerability rewards program ("bug bounty") for the open-source release of Kubernetes. What do we hope to get out of this?

- Attract security researchers to get more eyes on the code, shake out security bugs, and put money behind k8s security guarantees
- Simplify K8s' security team's security bug triage and response
- Demonstrate that K8s is "enterprise ready"
- Allow all contributors to participate in improving K8s' security, while remaining independent of any single K8s contributor

The aim would be to announce this new Kubernetes bug bounty program at KubeCon EU in May 2018.

This should NOT replace or interfere with existing vendor-specific bug bounty programs for their deployments of Kubernetes, e.g., if a bug is in the core Kubernetes components, it should be reported to/ routed to this new program; whereas if a bug is in Google's specific implementation of Kubernetes in Google Kubernetes Engine, it should be reported to/ routed to the Google Vulnerability Rewards Program.

Scope

To run a successful bug bounty program, we will need a well-defined boundary for what's in scope and out of scope in terms of the 'core Kubernetes components'. This should also account for best practices that are not necessarily on by default in Kubernetes, but exist to prevent common security vulnerabilities.

We would need to formally define the scope and vulnerability taxonomy (i.e. reward for each type of attack), likely with input from the bug bounty program provider. Some considerations:

- **Versions**: the intent would be to specifically target vulnerabilities for versions that enterprises are reasonably running in production. This could be, for example, the last three major releases, on a rolling basis.
- **Stable features**: the intent would only to include Stable and Beta features, not also features in Alpha.
- **Best practices**: the intent would be to include all recommended security features in Kubernetes, including features like:
    - RBAC, and disabling legacy authorization
    - Remove default service account permissions
    - Network Policy with the appropriate default policies
    - Pod Security Policy
    - seccomp and AppArmor
    - Encryption of secrets at rest, not using the 'identity' secret provider
    - Disabling the Kubernetes UI
    - etc.

  Some starting points for this list of 'best practices' include:
    - https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/
    - https://schd.ws/hosted_files/kccncna17/d8/Hacking%20and%20Hardening%20Kubernetes%20By%20Example%20v2.pdf
    - https://cloudplatform.googleblog.com/2017/11/precious-cargo-securing-containers-with-Kubernetes-Engine-18.html
    - https://coreos.com/os/docs/latest/hardening-guide.html
    - https://www.cisecurity.org/benchmark/kubernetes/

The work to define the 'best practices' for Kubernetes security would essentially boil down to expanding on a Kubernetes hardening guide, and make this easy and repeatable to deploy, e.g., by sharing an example app. It would be beneficial for the community to take this on as part of this effort, and publish and maintain these recommendations as part of Kubernetes documentation.

A related conversation/ idea is to develop a 'security conformance' for Kubernetes, similar in concept to the Certified Kubernetes conformance program. For a given set of best practices/ hardening guide, a 'security conformant' distribution would implement some minimum set of these practices. This is not in the scope of this document, but would benefit from the same hardening documentation effort.

Operations

Current security incident response

Currently, Kubernetes has a security bug triage and disclosure process, detailed here, as follows:

- A researcher reports a security vulnerability to kubernetes-security@googlegroups.com, which goes to the [Product Security Team](), a small number of engineers from various organizations. The oncall triages and responds to the bug appropriately, with an SLA of 3 working days, which may include:
    - Recognizing this as a new, critical unaddressed issue. At this point, the Product Security Team works with the researcher to identify a fix and a disclosure timeline. This is communicated to kubernetes-security-announce@, and once a fix is made available, may first be distributed to kubernetes-distributors-announce@.
    - Recognizing this as a new, unaddressed issue, which is not critical, and filling an issue in the Kubernetes github, e.g., https://github.com/kubernetes/kubernetes/pull/58720
    - Recognizing this as a known issue, and pointing the researcher to an existing issue in the Kubernetes github.

Note that the security vulnerability process currently treats all vulnerabilities as triggering a fix and release process, although the release timeline can be slower for lower severity bugs.

Proposed new security incident response with bug bounty program

The purpose of introducing a bug bounty program is not to revamp the security incident response process, but to (a) fit into the existing process as easily as possible, and (b) simplify triage, response, communication, etc. tasks where possible as the expected number of reported vulnerabilities will increase with having such a program.

If a Kubernetes bug bounty program were to exist, the security bug triage and disclosure process would be slightly different:

- A researcher would report a vulnerability to the third party bug bounty program provider, with information requested as per the bug intake form.
- The third party bug bounty program provider would do initial bug triage, including verifying validity and assigning the bug a priority. Based on the priority, it may
    - Trigger an email alert to the Product Security Team oncall for response, and to trigger the existing security vulnerability response process.
        - *In this case, the main difference is that information on the vulnerability would have an initial intake via a third party provider's bug bounty platform, vs. the K8s Product Security Team Google group. This information can still be sent to the Google group for posterity or safe keeping if that is the preferred outcome.*
    - Autofile an issue in the Kubernetes github component, and autoreserve a CVE.
    - Reject the issue as an existing known issue, with a bug.
    - Reject the issue as a provider-specific issue, and forward the information or redirect the researcher to file the bug as part of that provider's bug bounty program or disclosure process, e.g., Google Vulnerability Rewards Program for vulnerabilities found in Google Kubernetes Engine.

Costs

Funding

Offering a Kubernetes bug bounty program would have two main types of costs:

- Rewards for the researchers who find vulnerabilities
- Costs to run the platform and maintain the program, e.g., using a third party vendor. The use of a third party, rather than an existing program, is preferred in order to keep Kubernetes security bug triage independent of any single Kubernetes developer or distributor.

Rewards structure and payment processing

Depending on the severity of the discovered vulnerability, a financial or swag reward would be paid out to the researcher, with financial range to be determined. The vulnerability severity and reward amount would be determined by a vulnerability taxonomy that is publicly published in Kubernetes documentation.

Discussion questions

To be discussed at sig-auth on Feb 21 2018:

- Overall, do we want to pursue a bug bounty program for Kubernetes? What concerns does the community have?
    - There is likely to be an uptick in reported vulnerabilities, especially in the first ~6 months. Are we prepared and willing to address these?
    - Are there any concerns with using a third party provider for initial triage?
- Is anyone interested in contributing to scoping the program, and/ or writing an example application for use by security researchers?
- Is there any special preference on the vulnerability taxonomy that is used? Else, we will start with a generic one, tailor it for Kubernetes, and distribute to sig-auth for any further feedback.
- What is the SLA requirement for a third party to triage bugs? Given that the current Product Security Team promises 3 business days until a response, how should this be changed? Should there be different response SLAs depending on the severity of the bug?
- What components outside of k/k should be in-scope?
- Can we realistically achieve a 90-day report to public disclosure deadline in Kubernetes?
- When should we announce it publicly? Kubecon EU during the sig-auth update?
- Is there also interest in joining the oss-fuzz project for continuous fuzzing and detection of vulnerabilities? Is the K8s Product Security Team willing to look at any bugs arising from this as well?