Inside the CNCF Project Security Reviews

Justin Cormack

Who am I?

Lead Security Engineer at Docker

Contributor to CNCF SIG Security

A maintainer of CNCF Notary project

@justincormack

CNCF security reviews?

CNCF security audits

- The CNCF requires all graduating projects to “have completed an independent and third party security audit with results published and all critical vulnerabilities need to be addressed before graduation.”
- The CNCF is offering to pay for these audits for any project, including even sandbox projects!
- These would normally cost lots of money.

Which projects have been reviewed?

- Envoy
- CoreDNS
- TUF and Notary
- Prometheus
- containerd
- OPA
- NATS
- and (ongoing) Kubernetes

Who did the reviews?

- Cure53, based in Germany performed all the published reviews
- Kubernetes is being performed by a collaboration of Trail of Bits and Atredis aka “Trail of Atredis”
- each audit typically takes a period of a few weeks
- typically start with code review, follow with guided black box testing
- around eight people involved per review, many eyes...
- tools used include modified clients and servers to check protocol error handling, fuzz testing
- also examine common attacks for the class of protocol, and specification edge cases

CNCF Security working group

CNCF SIG Security

- formerly the SAFE Working Group
- now the official CNCF SIG Security
- starting to prepare *security assessments* for projects
- these are *not* pen tests, but are outlines of security relevant parts of the project, and what you should to know before using
- we are working through projects, starting with the security focused ones, including both existing and proposed
- all are welcome to participate

Ok just give me the results which is the most secure?

This is not a competition, it is a process

- overall, all the projects reviewed so far did pretty well
- the details are actually more interesting than the summary
- excludes vast areas of issues not found
- oh, ok let us summarize...

Summary

	Vulnerabilities				Issues
	critical	high	medium	low	
CoreDNS	1				3
Containerd					1
Envoy		1	3		4
NATS	1		1	3	4
Notary/TUF			1	1	2
OPA		1	1		3
Prometheus			1		1

Common themes

Common themes

- it is much harder to write bad Go in modern high level languages than in C
- most of the projects reviewed so far are written in Go and have few issues
- the NATS C client caused most of the NATS issues
- Envoy's C++ code had some similar issues
- these can and were mitigated, see later...
- scope issues were important, more later...

CoreDNS

CoreDNS most fun (critical) vuln

- The CoreDNS application allows to configure the caching of the DNS responses via the cache plugin. It was discovered that CoreDNS only verifies the transaction IDs but fails to check whether the domain in a request matches the response. This can be abused to inject malicious A records in the cache of the DNS server.
- *Moral:* validate, validate, always validate anything from the outside world

Biggest takeaways

- CoreDNS uses an excellent external Go DNS library miekg/dns
- “There is little doubt out there that infrastructure of this size and complexity, implemented in modern environments like Go, is frustratingly hard to audit, whether it is subjected to manual static code analysis or actual dynamic pentesting.”
- “It is strongly recommended for the CoreDNS team to fully integrate security-related auditing into the development process, as auditing as an afterthought simply does not suffice.”

Containerd

Containerd most fun ~~vuln~~ issue

- no actual vulnerabilities found, and only one issue
- It was discovered that a malicious configuration file makes it possible to create a persistent arbitrary directory on the filesystem. What is more, the owner and group ID can also be set arbitrarily in this context.
- Hard to see this can be exploited as you probably need to be root anyway to configure this, hence an issue not a vulnerability.

docker

Biggest takeaway

- “While the scope of this test and the architecture of the system were rather well-defined, a threat model was not communicated to Cure53 by the development team. Therefore, it took more effort than originally expected to delineate an adequate attack surface.”

NATS

NATS biggest takeaway

- a simple protocol means less to go wrong
- safe languages such as Go and Java have few issues
- “The small caveat should be applied to the C implementation, which definitely calls for some additional care and security-investment before being in a state that is safe enough for public consumption.”
- of the vulnerabilities
  - 1 critical, 1 medium, 1 low in the C NATS client
  - 1 low in gnatsd
  - 1 low in node-nats
- similarly with the issues, half are in cnats

docker

The price of C is eternal vigilance

“Cure53 found that the provided client library examples were minimal in their implementations, therefore exposing very little attack surface as far as code auditing and penetration testing was concerned. However, one “Critical”-ranking finding in the C implementation of the client, paired with a few less severe findings in this particular client, gives Cure53 some cause for concern about the state of security in this item.”

If you must write C code

- consider using something else, eg Rust can produce C ABI
- or consider writing multiple native bindings
- even a small simple client library is a potential minefield
- understand how much work is involved
- write for safety first
- use static and dynamic analysis tooling
- use ongoing fuzz testing
- portability brings further problems
- avoid parsing where possible
- get security audits

Envoy

Envoy biggest takeaway

- Admin interface wasn't secured
- Accepted commands from GET requests, with no CSRF protection
- Could shut down Envoy
- Admin interfaces are often an afterthought or convenience, please just omit it if you are not going to harden it.

Does C++ suffer the same problems as C?

- in a few cases where you use it in C style, yes it can
- a small number of C style issues such as potential use-after-free and int32 overflow were found
- the more you use modern C++ the less this is an issue
- “the team run various code quality assurance tools and scanners to make sure that the software was not affected by the bugs and memory corruptions commonly affecting similar projects”

TUF and Notary

TUF and Notary biggest takeaway

- both projects had previously had security audits
- audited together as Notary is an implementation of TUF
- despite trying to give the team hints of things that might be an issue they could not find anything significant
- most interesting issue is an input validation problem

docker

Open Policy Agent

OPA biggest takeaway

- OPA query interface vulnerable to XSS
- XSS issues in any web exposed interface are a common theme
- this one from unsanitized input
- never forget to be paranoid of untrusted input!

OPA takeaway 2

- confusing documentation issue
- you can set an authentication token in the config
- but you also have to enable authorization in the config!
- always try to error or do the least surprising thing for config, or design your config language to be less surprising

Prometheus

Extreme scope disagreement

- “Note: This was flagged as a false alert and an expected behavior by the Prometheus team.”
- “The results of this Cure53 2018 assessment of the Prometheus software compound are rather mixed. This stems from varying perspectives on security represented by the maintainers of Prometheus vis-à-vis the Cure53 testing team. On the one hand, the overall indicators were very good with low number of findings, strong and clearly-written code and very well-chosen and properly-deployed security properties. On the other hand, the testers are concerned with the general assumption about the removal of security as the noteworthy realm by shifting this task to the outside layers and parties.”

If you cannot agree scope, do not audit

- the only real outcome was a docs PR https://github.com/prometheus/docs/pull/1063
- there are good arguments that some projects may not want to cover all of security
- who is responsible for hardening a component? Upstream or user?
- are there useful things Prometheus could do to help all users or are use cases too different?
- legitimate scope for disagreement, but essentially a waste of money and time doing an audit without agreement up front.

My project wants a review what should I do?

If your CNCF project would like a review

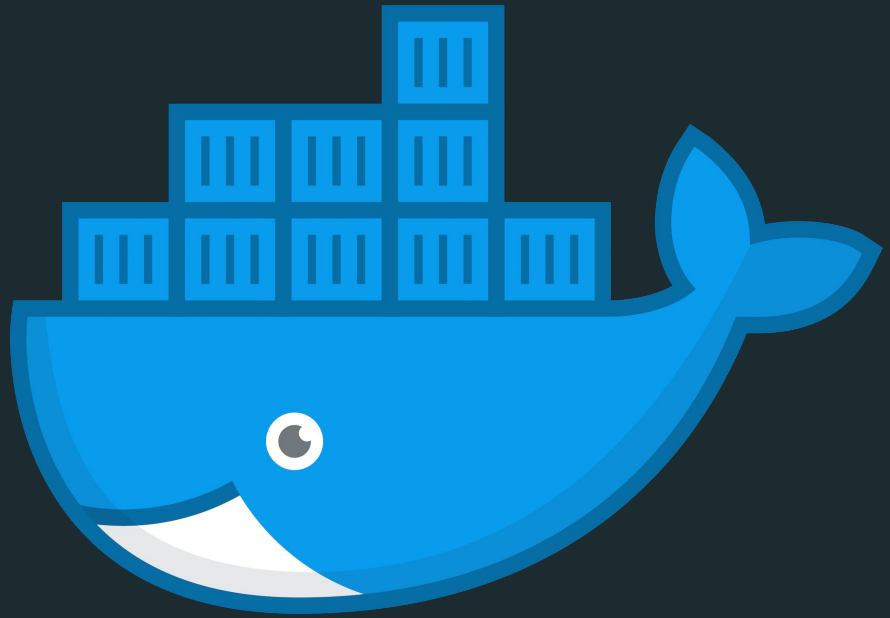
- the process has not been formalised fully
- come and talk to SIG Security

Summary

In summary...

- understand your threat model first
- come and talk to https://github.com/cncf/sig-security if you want help
- document what users need to know about security
- strengthen your code
  - do not trust inputs
  - careful with that C
  - tooling can help with common issues
  - deep dive on security critical parts
  - review, review, review
  - keep a list of parts you worry about
- external reviews are part of your security journey

Questions?



THANK YOU