Practical Platform Security
A No-Nonsense TODO List
for
Product Development Teams
and Leaders

Version 1.5

Background and Introduction

# Introduction

This document is created as a TODO/CHECKLIST for a security conscious product development team that is developing and deploying a product or service.

This document assumes a certain SaaS based service architecture, which is ubiquitous in this day and age. Several SaaS service deployments are done in the public clouds, such as GCP, AWS, Azure. A good percentage of saas services are deployed and operated out of private data centres / private clouds. While a lot of services continue to be deployed on virtual machine appliances there has been a major shift into deploying using a microservice kubernetes based architecture. This paper will consider that as the de facto deployment model .

The responsibilities of addressing each item listed in this document may fall on different parties in the larger organisation. The platform or the operating system runtime system may address most of the table-stakes items, however this will vary based on the provider. And in case of Kubernetes being deployed by the product team itself this list becomes very useful. Even if the platform provider supports several of these, engineering teams and leaders should be aware and should know how a certain security topic is handled and supported by the platform.

Note that this paper does not attempt to address application security. No discussion is provided on each topic - a later version of the document may provide additional details, and possibly points on HOWTOs for each topic or checklist item.

In addition to security, this document maay contain a set of items that overlap with general saas/cloud deployment best practices.

This clearly is not an exhaustive list, but a very good starting point for buttressing your security. None of this is theory - all of this has been done/deployed used at some point by the author.

A JIRA import CSV version of this list, with priorities categories, and tags can be made available upon request.

General

- Remove any and all make/model/version numbers from login pages, unauthenticated public access pages, login prompts, network issue texts of all third-party software that you may be using.
- Stop checking-in SECRETS/PASSWORDS/Certificates/Private Keys into your source code repositories. Use passbolt/vault or equivalent.,,General,1,PracPlatSec Task,PLATSEC,Implement or enable audit and access logging for all systems that allow a login."
- Run separated Dev, Staging, Production environments, preferably with no direct network connectivity between them.
- When using self signed certificates, Create and use separate root CAs and corresponding trust-chains for each environment and do not trust development, staging signers/CAs in production and vice versa.
- For non-browser client applications that connect to your services always use certificate key pinning. That is to say, build-in a set of trusted CA public keys in your application code and ensure that the CA chain is used for all connections at your service. You may need more than a couple of keys in case you need to rotate the certificates or decommission CAs.
- When connecting to external third-party services, enable OCSP (Certificate Status) checks on the client pods, software, wherever possible. In programmed code implement/enable OCSP checks. Check your third party software configuration and enable OCSP

Thirdparty and Opensource

- Disable all automatic software/module updates from internet third party / open source repositories.
- Establish a process for periodic review of third party CVCEs and updates to this software modules.
- Ensure that your build system can complete a full build and produce artefacts without any access to the internet. This may sound oxymoronic to modern programming paradigms, and this may also mean maintaining copies of the right forked versions of third party or open source code inside your corporate repos and keeping them up to date.
- Build all open source third party code from labelled sources yourself. Try to avoid using pre-built binaries, wherever possible.
- For all open source / third party software images/containers deployed - always maintain the version numbers easily accessible along with their licences. This should include anything pulled by imports such as - go imports, maven imports, npm, reactjs, vue-js dependencies etc.
- Ensure your build system can produce EXACTLY the same artefacts for every build, if there are no commits to your codebase and no changes to the build system (AKA ensure there are no unintended pulls/upgrades to imported, third party modules, dependencies during the build process)

Infrastructure-DNS

-

Avoid unintended DNS leaks, Ensure all internal names are resolved internally. Avoid DNS poisoning issues.

- Avoid DNS leaks in development servers and development PCs also.
- Stop configuring public DNS servers like 4.4.4.4, 8.8.8.8 etc on all your vms, pods and container images. Allow the containers to use the k8s-DNS. Allow k8s DNS to point to a filtering outbound DNS proxy centrally in your cluster. Manage the external DNS that you want to use centrally.
- Enable category, IP, reputation, and geography based DNS filters on your filtering DNS proxy.
- Cloudflare and several other vendors offer services that cater to this need
- Add Certification Authority Authorization CAA records on your DNS for your domains where they have publicly exposed services that your client applications connect to.
- Consider adding DNSSEC records for your domain name - if you're a web-based service.
- Ensure SPF and DMARC records are configured properly on your DNS. Ensure DKIM keys are configured on your system.
- Update DNS configuration to limit recursion/forwarding.
- Limit the subnets/source Ips that an internal DNS server responds to
- Disable DNS 'updates' where not necessary.
- Have an automated/robust process to renew DNS domain names.
- Where possible use an internal NTP Secondary within your cluster.

Servers

- Turn off automatic software updates on all production servers. But selectively enable only automatic security patching on your systems.
- Deploy endpoint monitoring agent and a syslog forwarding agents on all your servers and ensure they start at boot time and they actively send data out to a central monitoring system.
- Disallow SSH forwarding, disallow root logins from remote (Use console), disallow empty passwords, Remove any login prompt/issue text exposing OS and kernel version on your servers.
- Disallow/Do not accept any old ciphers such as CBC (instead use AES+GCM) and old hashes such as MD5, SHA1 etc in the SSH, Webserver TLS configurations. Ensure this is done in programmed server/client code also.

Kubernetes

- Stop exposing kubernetes control planes and dashboards of dev/test systems to everyone on the internet. Use VPNs via a bastion host to your control plane. Use IP Whitelists.
- Enable HTTPS for kubernetes API server access for development, staging, production.
- Allow only authenticated access
- Allow only a small set of well known IP addresses that you control as WHITELISTED/Allowed source IP address for the

kubernetes api-server access,.

- Automate configuration of this Source IP whitelist for kubernetes API IP addresses on nodes running your kubernetes.
- Ensure Source IPs in the WHitelist are from a RESERVED IP pool if using public cloud.
- Add source IP WHITELISTS for any Kubernetes dashboard software for kubernetes.
- Ensure that all kubernetes service configuration files, configuration files, run time data directories in the master and slave nodes are only read/writeable by the owner. Normally this owner will be 'root' user.
- Implement role based access control on your kubernetes cluster, allow only explicitly authorised users to access functions/apis.
- Disable debug, exec, interactive shell etc on your production pods.
- Do not share kubernetes keys/tokens for API/ CLI access across users, use separate user identities with different roles. Where possible avoid using permanent tokens, use sso logins for kubernetes API access. Enable token expiry.
- Ensure any kubeconfig files are readable/writable only by owner (in most cases case root user). These may contain tokens.
- Ensure that kubernetes is configured to lookup service accounts when authenticating with tokens always. This may slow down requests a bit but handle deleted accounts before expiry properly.
-

Integrate kubernetes API server authentication with your enterprise directory using Open ID connect / Oauth / SAML mechanisms and define proper access group/role/policies on the directory provider/identity provider server.

- Enable kubernetes API rate limiting and configure limits to a value that is acceptable for your normal usage.
- Implement or enable audit and access logging kubernetes API servers. Implement proper log rotation, compression and archival and purge policies wherever logs are generated.
- Enable Encryption at rest for Kubernetes confidential data such as kubernetes Secrets. Your platform provider may provide KMS systems and encryption providers to achieve this.
- Separate logically different functions into different kubernetes namespaces with no connectivity between them.
- Avoid running production, development, staging containers on the same kubernetes cluster. Use separate kubernetes clusters for this purpose.
- Implement a kubernetes validation/admission/policy hook to minimally validate atleast the 'source' base URL for container images. This will ensure that your kubernetes instances only pulls containers from KNOWN URls and there are no accidents.
- For sensitive private container images use AlwaysPull policy so that the cached container images cannot be used to start pods by others on the node. Note that this may slow down startup of containers.
- By default do not allow users to expose external Load Balancers to their pods, allow them to use only ClusterIP or

node IP.

- Implement a process for upgrading kubernetes itself, or test this process if using public cloud.
- Implement/Enable audit logging of all Service-Account/Token usage and send these to the centralised logging/event management system.

Kubernetes Nodes

- Implement/Enable Mutual TLS (client cert verification and server cert verification) between the kubernetes nodes by using client certificates on all nodes that connect to the master.
- Consider using your own CAs for your cluster and verify server and client certificates. Some systems do not allow a way to revoke certificates in these cases avoid using client certificates.
- Separate and tag/taint nodes as those having outbound internet access and those that do not.
- Disable node ssh and/or permit it only for a very select set of user roles.
- Stop using common/shared login accounts for remote shells/accesses.
- Where possible allow only whitelisted source IPs to be able to ssh into the host nodes.

Kubernetes Pods

-

Run pods that do not require internet on nodes with no internet only.

- Other than DNS,NTP ideally no pod should need any other internet access. If local subnet servers are used for DNS and NTP - then even this should not be required.
- If only KNOWN internet destinations are accessed by pods implement a forward proxy or outbound proxy or egress gateway that allows only those destinations or IP addresses.
- Implement Kubernetes NetworkPolicies, enable Egress and Ingress pod isolation by default. Prevent internet access or cross cluster access or cross namespace access for pods that do not require this. Use pod/namespace selector matches for these network policies.
- Implement Pod Security Policy for all pods, avoid defaults.
- Use least privileges for your pods
- By default disallow access to host network, host ipc, and host process namespaces for pods
- Avoid Host path access to pods wherever possible.
- Ensure allowedCapabilities property is set properly and limited to only what the pods need. Explicitly requesting required minimal set of capabilities is always better than using defaults.
- Ensure all Pods have CPU and memory resource limits where practical to avoid them overwhelming the cluster.
- Ensure all pod logs are to your favourite centralised monitored logging/alerting system, ensure these logs are sent securely. (syslog tls etc)
-

Implement Mutual TLS (Client cert and Server cert verification) between the applications running on all pods even if the communication is internal.

Container Images

- Implement separate Dev and Production Artefact/Container Registries.
- Sign container images/image hashes with specific private keys and trust only those in your operations code. Secure these keys.
- See if GKE binary code authorization or a similar mechanism is useful
- Implement user authentication for Push and Pull requests on your registries.
- Implement separated keys/tokens/service-accounts based authorization for push requests and for pull requests on your registry.
- Implement different registry access keys/tokens/service-accounts across Dev, Staging and Production
- Ensure that there is a mechanism in place to invalidate static keys/tokens that are issued to systems/or users for API access.
- Remove any make/model/version displays on the registry control panels, login pages.
- Separate Code/Manifests, Config/Cluster-specific-values, Secrets/Credentials wherever possible.
-

Remove all SECRETS/PASSWORDs/Private Keys from container images. Even for dev and test images.

- Deliver Passwords/Secrets/Certificates/Private Keys to a container only via a keyphrase store such as an external Vault or local kubernetes secrets/sealed secrets (also see kubernetes encryption at rest).
- Ensure that only a selected list of people with the right authorization are allowed access to view kubernetes secrets.
- Remove any registry / push pull tokens/secrets into any container images.
- Run VA scans such as Snyk on all your container images, continuously - as part of your CI/CD cycle.
- Avoid exposing registries over the internet, use Private 'internal' IPs to pull from registries.
- Run scans during regular development on a daily basis on your containers
- Scout, Snyk, Google Vulnerability Scan etc
- Disable Auto-update, motd, release check and such services from your container images.
- Ensure all container images have process starts that will run as a non-ROOT user (Run as USER tag), unless absolutely necessary otherwise.
- Consider having a trust chain (ca-certificates) on all container images you build that contains only the certificates signed by your private CA. This will ensure that the pods will not connect to any man-in-the-middle server posing as your service.
-

For all other cases keep the ca-certificate chain regularly updated and secure. (ubuntu update-ca-certificates)

- Use the 'USER command in your DOckerfiles. Avoid running processes as root unless absolutely required."

Applications

- Implement Mutual TLS Server Authentication and Client Authentication even for internal pod to pod communication.
- Only allow 'intended' client applications to be able to connect to server applications, (client cert/client subject name based authorization)

# Appendix

# Appendix A - Kubernetes server options to review.

Kubernetes server and the kubelet commands provide a lot of options to fine tune security related configuration. This configuration needs to be reviewed in every kubernetes environment. A set of security related options are provided here for review - always refer to the latest kubernetes documentation

Reference:
https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

As of March 2024.

```
--anonymous-auth
--audit-policy-file
--authentication-config
--authorization-config
--authorization-mode
--authorization-policy-file
--authorization-webhook-config-file
--bind-address
--client-ca-file
--cors-allowed-origins
--enable-admission-plugins
--encryption-provider-config
--etcd-cafile
--etcd-certfile
--kubelet-certificate-authority
--kubelet-client-certificate
--oidc-ca-file
--oidc-signing-algs
--peer-ca-file
--root-ca-file
--service-account-extend-token-expiration
--service-account-key-file
--service-account-max-token-expiration
--strict-transport-security-directives
--tls-cipher-suites
--tls-private-key-file
--tls-sni-cert-key
--use-service-account-credentials
```

Revision History

Version	Date	Comments	Modified By
0.1	March 16 2024	Initial Draft.	srini@hackacharya.com
1.0	March 21 2024	Updated with more items.	srini@hackacharya.com
1.1	March 27 2024	Updated with additional items. Removed OS/DB empty sections for now,Added kube server options in the appendix.	srini@hackacharya.com
1.2	March 28 2024	Minor updates	srini@hackacharya.com
1.4	Aug 22 2024	Gen from CSV	srini@hackacharya.com

hackacharya@gmail.com