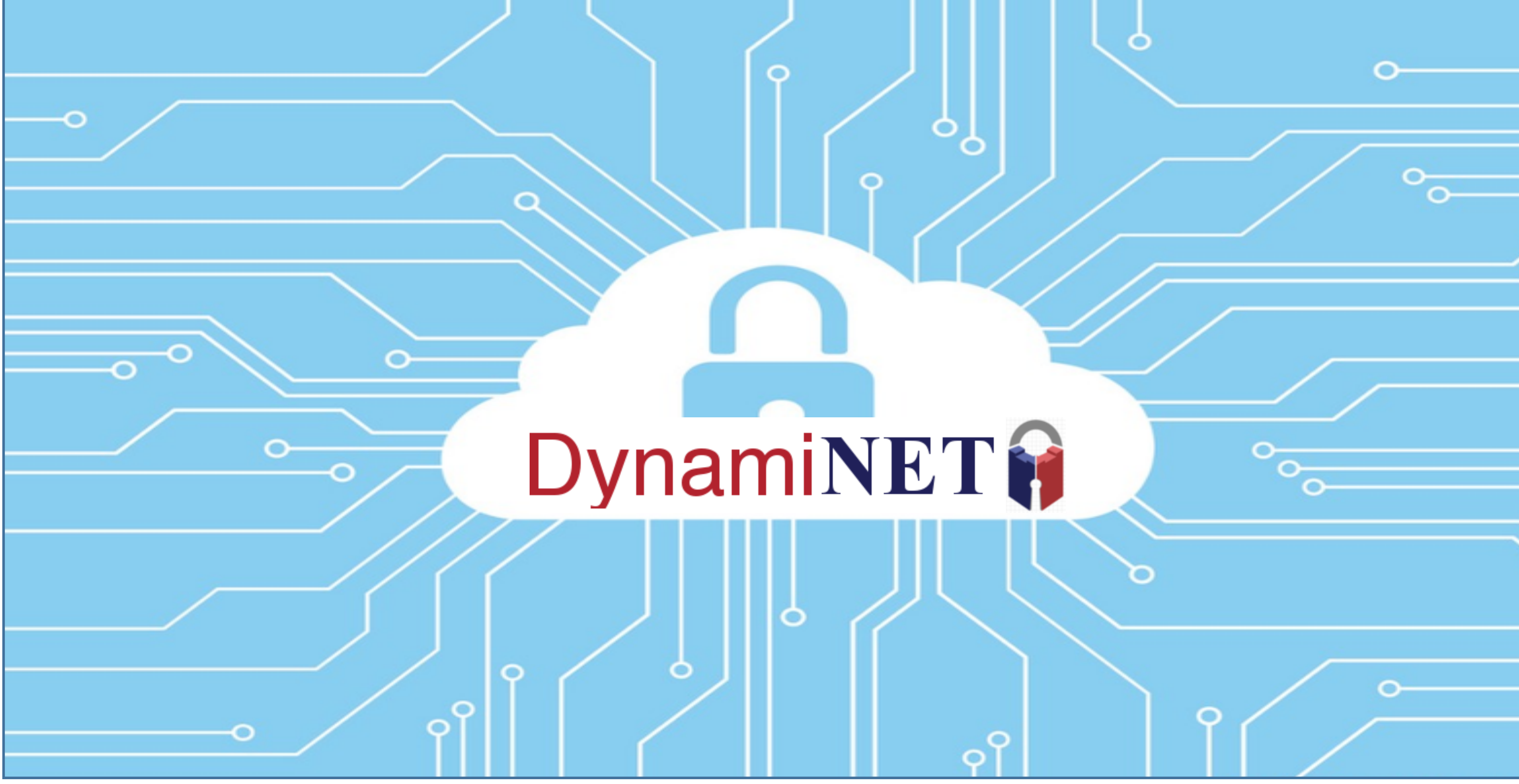

Cloud Security 101

September 2019

TABLE OF CONTENTS

Executive Summary

The cloud platforms, Amazon Web Services (AWS), Google Cloud Provider (GCP) and Azure, have become very popular in recent years. Many companies are moving away from the traditional on-premise metal infrastructure and turning towards scalable and cost-effective cloud providers. Likewise, delivery teams are leveraging these new platforms to host scalable, responsive and highly available services to accelerate the delivery of products to their customers. The adoption of application containers, orchestration and network layering tools to speed up the development and delivery to cloud platforms is accelerating. Security professional risk being left behind by this fast-paced evolution and, left unchecked, delivery teams risk introducing new vulnerabilities into their production environments.

There are many resources available to help cybersecurity professionals learn about these new technologies, such as AWS, Docker, Kubernetes and Istio. There are also many resources to ensure delivery teams are aware of the security elements of these new services. DevOps and cybersecurity teams must leverage these materials to enhance their understanding of these products and their associated security weaknesses.

Guidelines and standards have matured in recent years to provide useful sets of check-lists to assess the security of containerised applications delivered to cloud platforms.

Introduction

In the fast-paced world of software development, it is often difficult to keep on top of the latest advancements in technology. Every month it seems a new development stack or a new programming language is mentioned in the latest technology newsfeeds. Many of these ‘advancements’ are nothing more than words wrapped around an aged framework to pump new life into it. Others fall by the wayside as quickly as they appear – open-sourced but not popular. However, sometimes, the new words that jump out of those newsfeeds and blogs keep doing so. Then you hear people in the office start talking about them. Suddenly, you realise that this new ‘thing’ has started to gain traction and is being used by your clients (or your employers). Scrum teams are excited about this apparent evolution – developers have a new tool in their toolbox to play with, architects have a new way of integrating and re-using components and product owners have the opportunity to deliver new features to their customers faster. But this enthusiasm is often tempered by the discouraging voices emanating from cybersecurity stating, “You cannot use that, we don’t know how secure it is!”.

Over the past few years, the biggest shift in software delivery is the use of Cloud Technologies, specifically Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP). The key driver for this shift is to reduce the dependency on companies managing their own infrastructure. Cloud Services provide the flexibility to expand and contract the underlying infrastructure based on demand, reducing the need for redundant and costly on-premise servers. This is potentially good from a cybersecurity perspective because it allows delivery teams to focus on application security while the Cloud Service Providers manage the security within their platforms. Although Cloud Service Providers offer many security patterns and tools to help delivery teams with their security requirements, there are many areas in which the security landscape needs closer inspection.

Delivering software to Cloud platforms brings new opportunities for product delivery teams. However, it also brings new challenges for security teams. Containerisation of an application has gained popularity in recent years, leveraging new capabilities that Cloud platforms offer, in particular the ability to develop and deploy applications as fully configured and self-contained entities, blurring the lines between development and operations.

The use of containers such as Docker and orchestration tools such as Kubernetes has grown significantly since July 2015[1]. Indeed, Docker and Kubernetes are frequently used together (69% of Docker containers are orchestrated by Kubernetes[2]).

The accelerated growth in these technologies is evidenced by the popularity of events that cover container and Cloud technologies. The flagship Kubernetes conference, a collaboration between KubeCon and CloudNativeCon, and organised by Cloud Native Computing Foundation (CNCF)[3], was first held in San Francisco in 2015 and has since expanded into Europe and China. Attendees

[1] https://medium.com/@rdodev/saved-you-an-analyst-read-on-kubernetes-growth-2019-edition-d34a3e5a8755

[2] https://cdn.thenewstack.io/media/2018/03/40c0a560-chart-kubernetes-manages-containers-at-69-of-organizations-surveyed.png

[3] https://www.cncf.io/

have grown from 500 to an estimated 8000 at each event[4]. Google, Microsoft and Amazon also hold regular cloud-focused conferences that are attended by thousands of delegates each year. It is fair to say, that these are not ‘fad’ technologies – there is something substantial happening here.

The rapid growth in Kubernetes and Docker poses some challenges for cybersecurity teams who need to ensure that delivery teams are using these technologies securely. A quick glance at the CVE database identifies 34[5] Kubernetes vulnerabilities and 54[6] Docker vulnerabilities (as of July 2019). However, there are other cybersecurity risks lurking within these technologies, such as user misconfiguration, inappropriate access control and poor implementation processes. There are potentially others.

The key question for any cybersecurity consultant who needs to support these technologies is, ‘what do I need to know to support the teams using Kubernetes and Docker and how do I find out more about them?’ This paper is an attempt to answer that question. It focuses on AWS, Docker, Kubernetes and tools that support these products and services. However, the principles are the same for other technologies.

[4] http://www.voxuspr.com/2019/03/what-is-kubecon-its-past-present-and-future/
[5] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=kubernetes
[6] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=docker

Methods of Learning

The speed of change in IT can leave many of us behind and so, it is important to stay up to date on the latest developments. However, even in this fast-paced world, in which technology advancements can overwhelm us, Cloud platforms and containerisation have matured to a level where their adoption is gathering momentum. This maturity provides us with a focus on what is important to learn and support from a cybersecurity perspective. A number of products have emerged over recent years to establish a market dominance, which cybersecurity specialists should make their focal point when surveying the everchanging landscape.

This section provides guidance for cybersecurity professionals (which really is everyone, right!?) to gain the knowledge and experience to support delivery teams that are forging ahead with these new technologies.

Resources for learning

Before embarking on a quest to understand the security concepts of cloud-based technologies, it is important to learn as much as possible about these new technologies, how they work and how they integrate. There are various options available for delving into this subject matter.

Online

The best place to start is the documentation provided by the proprietor or by the open source community. These are normally available on vendor websites or through open source communities such as GitHub. Often, reputable industry experts write blogs or articles covering many aspects of these technologies. As well as written content, industry experts may also publish videos and podcasts covering these topics. Many vendor sites contain myriad resources such as whitepapers, case studies and blogs, which, although specific to their own products, are useful, nonetheless. The three main cloud providers each maintain well-structured online documentation on their respective platforms.

Books

A number of books have been published in recent years covering cloud and container technologies. Once printed, a book can become dated and inaccurate very quickly within a fast-maturing industry. Nevertheless, there are printed books that are written well enough to stand the test of time. Others are self-published and updated regularly with updated editions. Some have e-book amendments to supplement the printed versions and of course, some books are only available as e-books. Reviews can act as a useful barometer on how useful and effective a book is, as can recommendations from peers and colleagues.

Courses

There are many courses available covering a wide range of topics on cloud and container technologies. Courses can be expensive, especially those that are tutor-led, or free, such as self-paced tutorials offered online. Likewise, courses are sometimes generic or, conversely, very niche. So, it is important to shop around. Each cloud provider offers industry recognised certification which acts as a good incentive to complete their courses and take their exams.

Conferences and meet-ups

Attending conferences is often enlightening as experts talk about and demonstrate latest technologies and new methodologies. Attendances range from a few hundred to several thousand and topics can be in-depth or generic. Nearly all conferences are sponsored by key vendors in the industry, often giving away free samples, low cost subscriptions or demos of their latest offerings, which provide more opportunities to learning about their products. Localised meet-ups are usually attended by smaller groups of individuals who come to share ideas, challenges and solutions on a wide range of topics. These smaller meetings tend to offer real-world experiences far away from the glamour of cutting-edge conference talks.

Hands-on Practice

Cybersecurity professionals who are more technically capable should use their new-found knowledge to build labs using the latest technologies. Understanding potential vulnerabilities and identifying weaknesses to exploit is a great way to deep dive into the security of containers and Cloud platforms. Most vendors offer community or open course versions of their products to allow technically capable individuals to learn about their products before adopting them.

Digging Deeper

Within this section, we focus on the specific technologies: Docker, Kubernetes and AWS, as well as some of the tools developed to support these products.

Learn about Docker

In the case of Docker, it would be advisable to start at https://docs.docker.com and specifically, the 'Get Started with Docker' resources page[7]. This website covers the current version of Docker, although there is an archive covering older versions of the container tool[8].

There is also a highly rated and very readable book on Docker by Nigel Poulton[9] titled *Docker Deep Dive – Zero to Docker in a single book* (ISBN 9781521822807). Nigel is a leading name in the container community having authored a number of books and training videos on Pluralsight.

What is Docker?

From the documentation, you learn very quickly that Docker is a software that creates, manages and orchestrates containers[10]. In essence, it is a platform for developing, shipping and running applications. Applications are packaged into lightweight containers which run on a host. Each container contains the minimum resources required by the application to run on the host. By abstracting software from the underlying infrastructure, Docker allows delivery teams to ship, test and deploy code quickly to a production environment[11]. Docker uses a client-server architecture – the Docker client talks to the Docker server (Daemon) which is responsible for building, running and distributing Docker containers.

What are images?

An image is a read-only lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings[12]. A Docker image has instructions for creating a Docker image. It is normal practice to create layers of images that are stacked on top of each other and represented as a single object. Each object contains a cut down operating system and all the files and dependencies to run an application. Images can be created and pushed to a private registry or pulled from an existing public registry.

What about Dockerfile?

An important component of Docker is a file called 'Dockerfile'. This simple text file lists a set of instructions which Docker reads to build images. The document contains all the commands a user could call on the command line to assemble an image[13]. Dockerfile is potentially a security risk if

[7] https://docs.docker.com/get-started/
[8] https://docs.docker.com/docsarchive/
[9] http://blog.nigelpoulton.com/
[10] Poulton, Nigel *Docker Deep Dive* (2018)
[11] https://docs.docker.com/engine/docker-overview/
[12] https://www.docker.com/resources/what-container
[13] The full list of commands used in a Dockerfile are documented on the Docker docs website https://docs.docker.com/engine/reference/builder/

configured incorrectly. Therefore, it is important to understand the concepts, structure and content of this file in order to carry out security reviews on how images are configured.

What are containers?

The runnable instance of an image is called a container. Containers can be created, started, stopped, moved or deleted. Container is defined by its image in addition to any configuration settings applied when the container is created or started. Each container is isolated into its own workspace via a set of namespaces. This isolation provides a level of security for running containers and is important to understand from a cybersecurity perspective.

What are Image Registries?

An Image Registries is a service responsible for hosting and distributing container images and is often provided by a third party as a public or private registry (such as the default DockerHub, Google Container Registry or AWS Container Registry). Within these registries, images are pushed to and pulled from repositories. The repositories hold multiple images stored as tags. In order to push and pull images, a user would log into a registry, create or find a repository and upload or download the relevant image tag.

Image Security

There are obvious security measures that should be taken to secure images that are consumed within a production environment. It is important to know how to restrict access to approved registries, whether they are private or public and control access to images within the registries and to ensure developers use approved base images. It is essential to scan images for known vulnerabilities and have policies in place to prevent vulnerable images from being deployed to production. Images need to be trusted and this will involve cryptographically signing images and cryptographically verifying them at runtime. Finally, it is important that image configuration files (in particular, the Dockerfile) are security reviewed and any potential risks are mitigated[14].

Docker security[15]

Docker security builds on the Linux security technologies embedded in the Operating System. Therefore, Docker supports Kernel Namespaces, Control Groups (cgroups), Capabilities, Mandatory Access Control (MAC) and seccomp, and builds on these layers. Out of the box, Docker supports mutual authentication, automatic CA configuration and certificate rotation and encrypted networks among many other default security measures.

[14] Poulton, Nigel *The Kubernetes Book* (March 2019)
[15] https://docs.docker.com/engine/security/security/

Namespaces

A Docker container is an organised collection of namespaces, which provides the necessary isolation for each container. Docker uses these namespaces to provide isolated process trees (pid), network stack (net), root file-systems (mnt), memory (ipc), and hostname (uts).

In relation to security, namespace isolation prevents one container from seeing or affecting processes running in another container or in the host system.

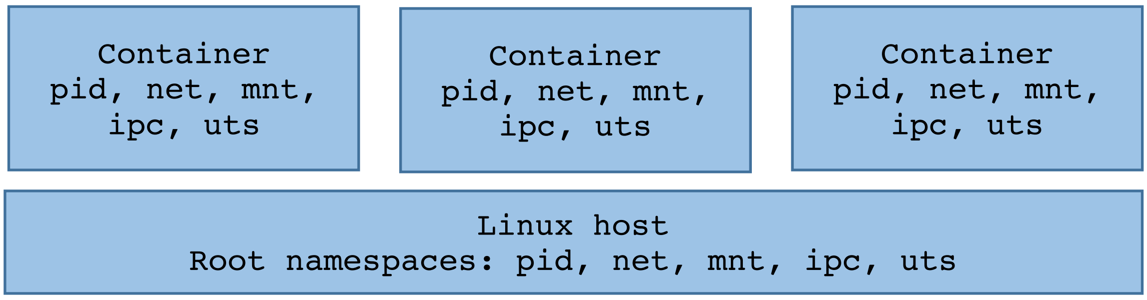
Container pid, net, mnt, ipc, uts
Container pid, net, mnt, ipc, uts
Container pid, net, mnt, ipc, uts
Linux host
Root namespaces: pid, net, mnt, ipc, uts

Figure 1 - Docker Namespaces

Control Groups (cgroups)

Although containers are isolated from each other, they share the same OS resources such as CPU, RAM and disk I/O. Cgroups sets limits on how much of these resources a single container can use. They are an important defence against some denial of service (DoS) attacks.

Capabilities

The Linux Kernel root account is made up of a long list of capabilities which provide granular control over what the root account is privileged to do. Docker makes use of these capabilities to allow containers to run with container 'root' privileges stripping out real root capabilities of the Linux host that are not required by the container. From a security perspective, it is best practice to remove all capabilities except those explicitly required for the relevant processes. In addition to this, it is possible to map the container root user to a non uid0 user outside the container, which mitigates the risk of breaking out of a container.

Mandatory Access Control (MAC)

Linux MAC technologies such as AppArmor and SELinux are supported in Docker containers via templates, providing an extra layer of security for containers.

seccomp[16]

Secure computing mode (seccomp) is a Linux Kernel feature that limits syscalls a container can make the host's kernel. Docker's default seccomp profile blocks a number of syscalls – the profile is a whitelist specifying the calls that are a container is allowed to make. Although it is possible to run a container without the default seccomp profile, it is good practice to ensure that the containers are run with the default profile.

Docker Content Trust (DCT)[17]

Docker security technologies provide an additional layer of security on top of the Linux security technologies. One of the more critical technologies in Docker is the Docker Content Trust (DCT).

[16] https://docs.docker.com/engine/security/seccomp/
[17] https://docs.docker.com/engine/security/trust/content_trust/

It provides the ability to use digital signatures to verify the integrity of downloaded images and signing of uploaded images. Enabling DCT acts like a filter, only allowing consumers to see signed images in the registry. There are three classes of keys used in image signing:

1) A client-side offline key that is a root of DCT and is used to create tagging keys. This key must be secured in a safe place
2) A client-side tagging key associated with an image repository that is used to sign image tags
3) A server-side timestamp key associated with the image repository provides freshness security for the repository

Within the Docker Enterprise Engine, DCT restricts users from using an image from an unknown source or building a container image from a base layer from an unknown source.[18]

Orchestration

Orchestration is used to manage and deploy applications, which means it provides up-scaling and down-scaling functionality, performs updates and rollbacks in an automated way. The use of containers has changed the way we deploy software and over the past few years the higher demands for managing and deploying containers has created the need for orchestration tools. The Docker tutorials walk you through the process of creating Docker Swarms[19] which is the native clustering engine for Docker. However, Kubernetes has become the main player in container orchestration due to a number of benefits it has over Docker Swarm. The key differences between the two technologies are: Swarm offers a simple solution to familiarise yourself with orchestration and is popular among developers who prefer fast deployments and simplicity; Kubernetes adds greater complexity and is widely used in production environments[20]. There are other orchestration tools such as Hashicorp Nomad and Apache Mesos(phere) as well as cloud native versions such as AWS ECS[21] and Azure Container Instances[22].

Learn about Kubernetes

Since Kubernetes has become the most widely downloaded orchestration tool for cloud-native production systems, it is worth taking a closer look at it. First of all, Kubernetes and Docker are complementary technologies – Docker is used to containerise applications and Kubernetes is used to orchestrate these containers. In fact, at the highest level, Kubernetes is a cluster (masters and nodes) for running applications and an orchestrator of cloud-native microservice applications. It is advisable to reference the documentation for Kubernetes at https://kubernetes.io and in particular start with the "Learn Kubernetes Basics" pages[23]

At a lower level, Kubernetes provides a framework to run resilient distributed systems taking care of scalability, failover and other requirements for production systems. Thus, Kubernetes provides:

[18] https://docs.docker.com/ee/ucp/admin/configure/run-only-the-images-you-trust/
[19] https://docs.docker.com/engine/swarm/
[20] https://thenewstack.io/kubernetes-vs-docker-swarm-whats-the-difference/
[21] https://aws.amazon.com/ecs/
[22] https://docs.microsoft.com/en-gb/azure/container-instances/
[23] https://kubernetes.io/docs/tutorials/kubernetes-basics/

- Service discovery and load balancing – for greater performance and stability
- Storage orchestration – for greater choice in storage systems
- Automated rollouts and rollbacks – for ease of deployments
- Automatic bin packing – for managing resources (such as RAM and CPU) for containers
- Self-healing – for reliability
- Secret and configuration management – manage secrets without exposing them within the stack configuration[24]

Simple security considerations for Kubernetes[25]

Kubernetes security is reliant on good configuration and best practices. Some key areas to focus on are:

1) Secure the communication with the API Server.
2) Secure the communication between Pods. It is also worth removing the Pods' access to the API server if the Pod doesn't require this privilege.
3) Secure Kubernetes components such as runtime binaries, images and configuration files while at rest (by restricting access, performing checksums, monitoring and alerting) and in transit (using TLS).
4) Enforce read-only policies on Pod filesystems.
5) Store key encryption keys outside of the Kubernetes cluster to protect the cluster store (etcd), preferably in Hardware Security Modules (HSMs) or cloud-based Key Management Stores.
6) Ensure configuration information used at runtime are stored separately from Pods.
7) Implement multiple authorisation modules, such as RBAC mode plus Node mode.
8) Implement a robust Pod Security Policy that prevents privilege escalation for an individual container. This includes:
    a. Force container processes to run as unprivileged non-root users.
    b. Ensure containers and pods have the right privileges.
    c. If supported by the Linux kernel, implement User Namespaces to map the container root user (uid0) to a non-root host user.
    d. Drop the root capabilities that are not needed to run the application in production.

What is a Service Mesh?

A service mesh is a dedicated and configurable infrastructure layer designed to extrapolate the away common network tasks from the application layer making service-to-service communication safe, fast and reliable. The abstraction of network communication from the application layer allows the scalability of microservices within a cloud-native environment since it allows developers to concentrate on application features and operations to concentrate on inter-service communication. A service mesh implementation uses a proxy server to allow one service to find another service with which it needs to communicate. This specialised proxy server is often called

[24] https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[25] Poulton, Nigel *The Kubernetes Book* (March 2019)

a sidecar. which runs alongside the container for which it manages the traffic flow[26]. The most common service mesh technologies include Istio and Linkerd.

Securing the Service Mesh

There are several security measures that need to be considered when working with service mesh technologies. Twistlock has produced a whitepaper with some high-level compliance checks for Istio[27].

Since the configuration of service mesh is managed using YAML, it is important to validate the YAML file for good security practice. In particular, ensuring traffic policies are enforced, Role-Based Access Control (RBAC) is in place, connectivity between services are protected (using mTLS), user roles are clearly defined, and service roles are bound to the correct processes.

Security policies can be applied at a granular level within the service, the namespace and the mesh to explicitly enforce access controls. The service mesh can manage certificates and keys to all namespaces and enforces different access control rules to the services[28].

Putting it all in the cloud

Cloud security is a vast topic, but it is important to understand the basic concepts and the security architecture associated with them. There are several cloud providers, of which the most widely used are Amazon Web Services (AWS), Google Cloud Platform (GCP) and Microsoft Azure. Each of the top three providers support Docker and Kubernetes, but also provide their own solutions.

Platform	**Build**	**Store**	**Run**	**Orchestrate**
AWS	AWS CodeBuild	Elastic Container Registry (ECR)	Elastic Container Services (ECS)	Elastic Kubernetes Services (EKS)
GCP	Container Builder	Google Container Registry	Container Optimised OS	Google Kubernetes Engine
Azure	Azure Pipelines	Azure Container Registry	Azure Container Instances	Azure Kubernetes Service (AKS)

Documentation for all of these services (and others) are available on the respective websites.

This paper does not attempt to examine the various security elements of native cloud container services. However, cybersecurity professionals would benefit from exploring the details of each cloud provider to gain a full understanding of the nuances that each brings to security. Just as importantly, as we have seen with recent exploits, some cloud providers are not inherently secure when using their default settings and potentially, misconfiguration of the service settings is likely to increase the risk of an attack.

[26] For more information on Service Meshes, the following resources provide definitions and solutions:

- https://www.nginx.com/blog/what-is-a-service-mesh/
- https://itnext.io/istio-service-mesh-the-step-by-step-guide-adf6da18bb9a
- https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/

[27] https://www.twistlock.com/wp-content/uploads/2019/02/Securing-Service-Mesh.pdf

[28] https://istio.io/docs/concepts/security/

Cloud Security

All three cloud service providers covered in this document offer a number of security features within their cloud platforms. Each offers variations and niche security services. However, they all offer the following:

- Infrastructure Security
- DDoS Protection
- Data Security and Key Management
- Identity Management
- Alerting and monitoring

They also provide premium professional security support for companies who prefer bespoke expert guidance and dedicated engineers to implement relevant security features.

Cloud Provider Whitepapers

All three top cloud providers have published many whitepapers on best practices for cloud architecture and cloud security. The following table is a guide to the current available whitepapers (as of September 2019):

Platform	**Security Whitepaper(s)**
AWS	https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf
Azure	https://docs.microsoft.com/en-us/azure/security/fundamentals/white-papers
GCP	https://cloud.google.com/security/overview/whitepaper

Focus on AWS Cloud Security[29]

Amazon offers a wide range of security capabilities: Amazon VPC native Network firewalls and AWS WAF helps control access to private networks in the cloud; Amazon CloudFront and auto-scaling helps protect against DDoS attacks; encrypted data storage, AWS Key Management Services and AWS CloudHSM help protect data stored in AWS; AWS Identity and Access Management (IAM) in addition to multifactor authentication help define and control access policies across AWS services; and, AWS CloudTrail and Amazon CloudWatch provide alerting and monitoring capabilities to reduce the impact of security events in a production environment.

Security Standards

There is a plethora of security guidelines freely available on the internet to support cloud-based solutions. They often provide a good baseline for minimum security requirements. These can be easily adopted in corporate security standards.

CIS Benchmarks

The Centre for Internet Security publishes and maintains a set of configuration guidelines for various technology groups. These CIS Benchmarks are designed to help teams protect their implementations against emerging cybersecurity threats. They are free to download. The following is a list of the CIS Benchmarks relevant to this paper:

Technology	**CIS Benchmark**

[29] https://aws.amazon.com/security/

Docker	https://www.cisecurity.org/benchmark/docker/
Kubernetes	https://www.cisecurity.org/benchmark/kubernetes/
AWS	https://www.cisecurity.org/benchmark/amazon_web_services/
Google Cloud	https://www.cisecurity.org/benchmark/google_cloud_computing_platform/
Azure	https://www.cisecurity.org/benchmark/azure/

NIST Security Guides

The National Institute of Standards and Technology (NIST) is a US based organisation established at the turn of the 20th century to foster innovation and competitiveness. Their Secure Systems and Applications division focuses on developing security solutions for high-priority and emerging technologies. There are several special publications that are of interest to the security professional working with cloud and container security:

Number	**Title**	**Location**
800-190	Application Container Security Guide	https://csrc.nist.gov/publications/detail/sp/800-190/final
800-204	Security Strategies for Microservices-based Application Systems	https://csrc.nist.gov/publications/detail/sp/800-204/final
800-144	Guidelines on Security and Privacy in Public Cloud Computing	https://csrc.nist.gov/publications/detail/sp/800-144/final

OWASP

The Open Web Application Security Project (OWASP) was established as a global community to provide open standards for secure software development. Their Container Security Verification Standard (CSVS) is a set of security controls for the development of container-based solutions. It is available as a PDF document and on GitHub via the OWASP CSVS resource page[30].

Known Vulnerabilities

Of course, as with most software, Kubernetes, Docker and Istio are not completely free from security vulnerabilities. Therefore, it is important to ensure delivery teams are using the most secure patched versions of the products. The following table covers the main products mentioned in this article and should be referenced regularly:

Technology	**CVEs**
Kubernetes	https://www.cvedetails.com/vulnerability-list/vendor_id-15867/Kubernetes.html
Docker	https://www.cvedetails.com/vulnerability-list/vendor_id-13534/Docker.html
Istio	https://www.cvedetails.com/vulnerability-list/vendor_id-19931/Istio.html

30 https://www.owasp.org/index.php/OWASP_Container_Security_Verification_Standard_(CSVS)

The Conclusion

Cloud-based technologies can appear daunting to those who are new to them. But there is a large selection of resources available in various formats to make the task of learning these new technologies very accessible. This educational material is particularly important to cybersecurity professionals who are often only exposed to traditional monolith applications running on local infrastructure. As DevOps teams adopt technologies to build, deploy and run their applications in the cloud, the challenge to the cybersecurity professional is to understand the security risks of these new services and new ways of working. Gaining knowledge and experience with cloud platforms, and the services that support application development and deployment, is important to the cybersecurity professional as the more traditional application architectures make way for new models.

Although many of the techniques used by cybersecurity experts to identify potential risks and vulnerabilities are still very relevant to cloud platforms, there is a need to adapt this skill to the services and processes that delivery teams employ. Cloud service providers often bake many security features into the products they offer. Yet, the responsibility of configuring applications, containers, orchestrators and services meshes is still with the delivery teams. Service settings, access control and network configuration can easily expose security weaknesses if not configured correctly.

In order to be able to support these delivery teams (which often work at pace), the cybersecurity professional must be able to adapt traditional methodologies to new technologies. But cybersecurity is a collective responsibility and the plethora of material available on cloud security is equally as important to the developer as it is to the security professional. It is important to make use of the resources available, both in terms of learning the technologies, as well as the security features of those technologies.

About Dynaminet

Dynaminet Ltd is a small independent company specialising in CyberSecurity, DevOps, Digital transformation, CICD and Delivery Process Management.

Email: info@dynaminet.co.uk