

A man with a beard, wearing a dark blue polo shirt, is looking up and to the right while holding a tablet. He is in a server room, with rows of server racks visible in the background. The lighting is dim, with some blue and green light emanating from the server equipment. The overall mood is professional and technical.

DARKTRACE

Securing and Investigating Containerized Applications Running on AWS ECS

Introduction

Cloud computing is no longer a new idea, but we are very much still in a period of transition for organizations who previously had all their workloads hosted on-prem, with many moving services into virtualized systems in public cloud providers such as AWS. Beyond just moving servers to be Virtual Machines (VMs) in the cloud, the architecture of services is changing too, with containerized microservices replacing monolithic software and the levels of abstraction between hardware running code and the end user is increasing.

The appeal of managed cloud services, which abstract away the server racks, are easy to understand (e.g., only pay for what you use, instantly scalable compute & storage, no maintenance of hardware, etc.), so more and more organizations are moving to the cloud.

One popular example of a managed service is AWS Elastic Container Service, or ECS. ECS can be configured to use EC2 (AWS virtual machines managed by the customer), on-premises container hosts connected to AWS, or AWS's fully managed Fargate as the underlying container host infrastructure.

This playbook covers best practices for securing and investigating containerized applications running in AWS ECS.

Best practices for securing AWS ECS

Understand the AWS shared responsibility model

The first step toward security in the cloud is to understand that security and compliance are a shared responsibility of AWS and its customers. To summarize, customers are responsible for everything “IN” the AWS cloud, whereas AWS is responsible for security “OF” the cloud. Detailed guidelines from AWS on this topic can be found [here](#).

Enforce a zero-trust Identity and Access Management (IAM) policy

Zero-trust security design follows the principle of: Never Trust, Always Verify. The concept assumes no default trust for any user. It enforces granular rules that define the scope of accessing data and workloads in the cloud, only allowing them to perform actions authorized within the defined organizational policies. Some recommendations include creating IAM policies in compliance with the zero-trust model (consider applying policy scope at the Cluster level), deploying onto the AWS ECS Clusters to isolate the infrastructure API from end-user access, and regularly monitoring how the AWS ECS API is accessed.

Ensure end-to-end encryption for secure network channels

Consider end-to-end [encryption](#) of mission-critical workloads running in AWS ECS environments. Encryption prevents unauthorized entities from being able to view or modify confidential information in transit. However, it’s important to note that this approach can add complexity in the event you need to investigate a potential compromise. Some additional things to consider are monitoring container network flows by using Virtual Private Cloud (VPC) Flow Logs to identify unexpected traffic to or from your ECS workloads and using separate AWS VPCs for workloads that require strict isolation.

Inject secrets into containers in runtime

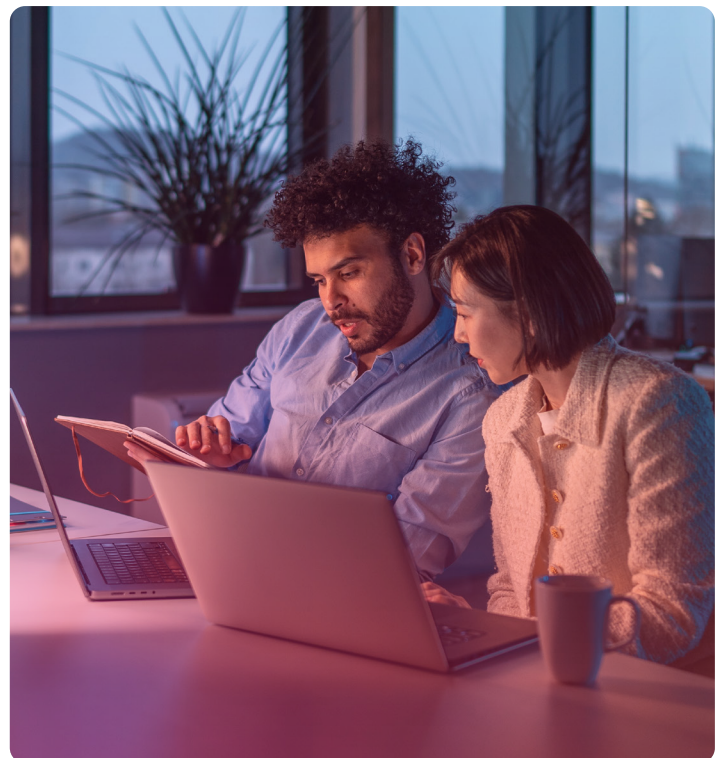
Follow a zero-trust security policy in managing the Secrets definition parameters for AWS containers. Secrets include login credentials, certificates, and API keys used by applications and services to access other system components. For example, use AWS Secrets manager for secure storage of Secrets credentials instead of baking them into the container.

Regulatory compliance as the bare minimum, not target security

Follow the guidelines of the applicable security regulations in your country and industry, but treat them as a bare minimum and not an end-goal of your ECS security plan. AWS helps deploy compliance-focused baseline environments especially focused on the following guidelines across most of the popular regulations such as HIPAA, PCI-DSS, and GDPR, among others. This includes understanding the shared responsibility model, enforcing strong IAM policy controls, leveraging the built-in tools and visibility provided by the cloud service provider, and having the processes and technology in place to investigate and respond to incidents that may impact your ECS workloads.

Gather the right data

Configure your container environments to communicate relevant security data and log data to the built-in AWS monitoring tools such as CloudWatch and CloudTrail. These tools can be used to collect data insights at the hardware, service, and cluster level. However, this data alone does not suffice for an in-depth investigation. More to come on this in the next section of this playbook.



Best practices for AWS Fargate

AWS Fargate is a serverless service that provides the option of fully managed and abstracted infrastructure for containerized applications managed using AWS ECS. The AWS Fargate service performs tasks such as provisioning, management, and security of the underlying container infrastructure while users simply specify the resource requirements for their containers.

The fully managed nature of Fargate is fantastic to reduce time and money spent worrying about failing drives or configuring operating systems, but this lack of infrastructure management comes with an equal lack of access. And this lack of access can add frustrations when it comes to security. Since AWS Fargate is a managed service, it offers no visibility or control into the underlying infrastructure. Due to this, it's important that you leverage a third-party solution to allow for data collection from running containers should you need to investigate one.

Construct secure container images

Container images consist of multiple layers, each defining the configurations, dependencies and libraries required to run a containerized application. Security of container images should be seen as your first line of defense against cyber-attacks or infringements facing your containerized applications. Constructing secure container images is critical to enforce container bounds and prevent adversaries from accessing the Host OS and Kernel. A few ECS container image security best practices that should be considered include using minimal or distroless images, image scanning for vulnerabilities, and avoiding running containers as privileged.



Incident readiness for containers

It's often said that incidents are 'when, not if' so preparing to investigate and respond is key, regardless of how robust your container security is. When investigating an environment which utilizes containers, data collection needs to happen quickly before automatic cluster scaling destroys valuable evidence. Additionally, you may have thousands of containers, so the collection, processing, and enrichment of container data needs to be automated. Some of the key things to consider are:

Automated data collection

Cloud scaling allows you to scale your investigation resources too, not just your applications. Collect and process container data in parallel triggered by an analyst or other detection or orchestration tool (e.g., SOAR) to get valuable information to analysts more quickly and reduce your time to incident containment and resolution.

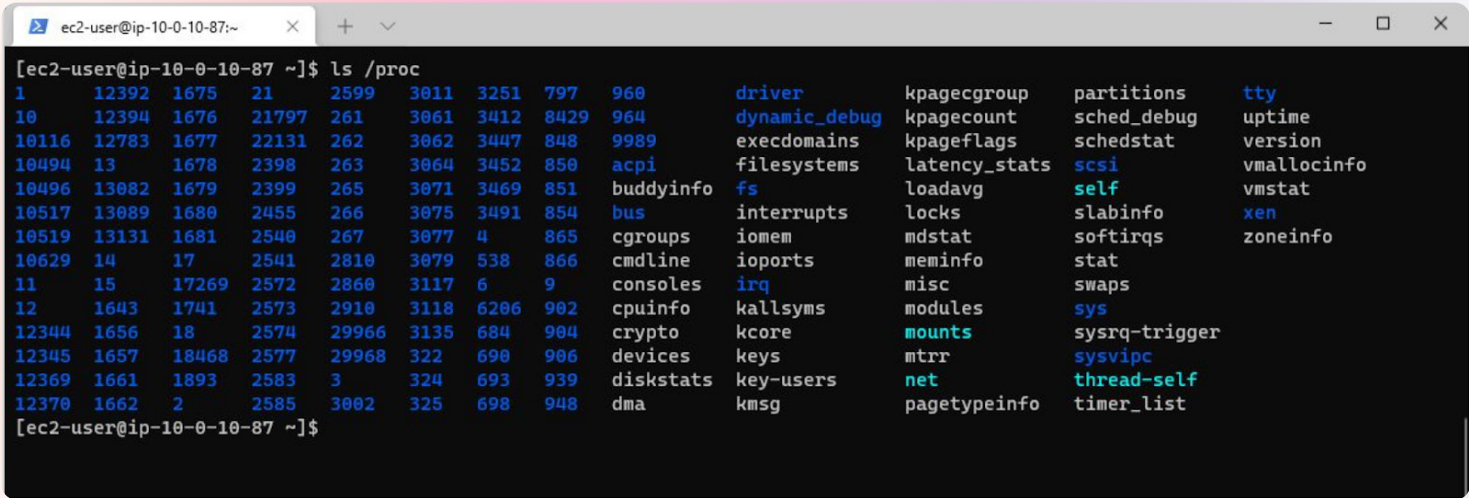
Level of visibility

While the visibility provided by built-in cloud service provider tools (e.g. AWS CloudWatch and CloudTrail) is important, these data sources alone are not sufficient to perform an in-depth investigation. Leveraging third-party incident and threat intelligence capabilities prove vital to gain a deeper level of visibility across container assets.

In this context, the useful data to include as part of your investigation are the system logs and files from within the container, the containers running processes and active network connections, the container host system and container runtime logs (if accessible), the container host memory (if accessible), and the AWS VPC flow logs for the VPC the container is attached to.

If data collection wasn't baked into the container declaration before the need to investigate arose, you need to rely on data you can actively interrogate out of the container. Or if you're using ECS on Fargate, taking an image of the running container or the container host isn't an option as the underlying AWS infrastructure is shared with no access to the end customers. In this case, the /proc directory gives us a snapshot of the volatile state of the container, much like a memory dump. Using this 'snapshot' we can build a basic picture of what is going on in the container at that time, such as running processes, active network connections, open files, etc.

This data provides a foundation of what you need to correlate with other data sources such as firewall logs, network subnet flows, etc. during an investigation to understand the actions carried out by an attacker.



```
[ec2-user@ip-10-0-10-87 ~]$ ls /proc
1      12392 1675  21    2599 3011 3251 797   968   driver      kpagecgroup  partitions    tty
10     12394 1676  21797 261   3061 3412 8429 964   dynamic_debug kpagecount   sched_debug   uptime
10116  12783 1677  22131 262   3062 3447 848   9989  execdomains kpageflags   schedstat     version
10494  13    1678  2398  263   3064 3452 850   acpi   filesystems  latency_stats scsi           vmallocinfo
10496  13082 1679  2399  265   3071 3469 851   buddyinfo fs            loadavg       self           vmstat
10517  13089 1680  2455  266   3075 3491 854   bus    interrupts   locks         slabinfo      xen
10519  13131 1681  2540  267   3077 4    865   cgroups iomem        mdstat        softirqs      zoneinfo
10629  14    17    2541  2810 3079 538  866   cmdline ioports      meminfo       stat
11     15    17269 2572 2860 3117 6    9    consoles irq           misc          swaps
12     1643 1741  2573 2910 3118 6206 902   cpuinfo kallsyms     modules       sys
12344  1656  18    2574 29966 3135 684  904   crypto  kcore        mounts        sysrq-trigger
12345  1657  18468 2577 29968 322  690  906   devices keys          ntrr          sysvipc
12369  1661  1893  2583 3    324  693  939   diskstats key-users    net           thread-self
12370  1662  2    2585 3002 325  698  948   dma     kmsg         pagetypeinfo timer_list
```



Container asset discovery

Since containers operate as virtualized environments within a shared host kernel OS, it's often challenging to keep track of asset workloads running across all containerized machines in a scaled environment. An agentless discovery process that can efficiently discover and track container assets can be used to enforce appropriate security protocols across all container apps.

Dedicated investigation tools and environment

If you need to figure out what tools to use and where to store incident data when an incident happens, you'll waste valuable time where more damage can be done and key evidence could be lost. Have a dedicated toolset or environment which automates as much of the investigative process as possible in place ahead of time so that you are ready to respond at a moment's notice.

Ability to quickly isolate

In the event a container is compromised, it's critical that you have the ability to quickly isolate it in order to stop the active attack and prevent further spread and damage. In some cases, isolation can be a good first step to take following initial detection. This will allow you to perform a more thorough investigation in the background and ensure proper remediation and containment steps are taken after you have a better understanding of the true scope and impact of the incident.

Conclusion

In summary, it's important to understand that the nature of AWS ECS as a managed service and the potential scale of an app running across multiple containerized environments makes it challenging to effectively capture data and investigate incidents. In addition to following the industry-proven AWS ECS best practices discussed in this playbook, a primary focus toward automation technologies is key to achieving a secure, high-performing, and defensible container environment.

How Darktrace can help

Darktrace delivers a proactive approach to cyber resilience in a [single cybersecurity platform](#), including cloud coverage. Darktrace / CLOUD is a real time Cloud Detection and Response (CDR) solution built with advanced AI to make cloud security accessible to all security teams and SOCs. By using multiple machine learning techniques, Darktrace brings unprecedented visibility, threat detection, investigation, and incident response to hybrid and multi-cloud environments.

In addition to detection and response capabilities, Darktrace can help security teams gain forensic-level detail, without forensic-level effort. Through the power of automation, Darktrace can enable security teams to capture evidence across server-based, container-based, and serverless environments with ease, including AWS ECS and even Fargate, across multiple AWS accounts.

Darktrace is a valuable tool for automating IR processes.

Learn more about Darktrace / CLOUD for AWS:

[Read the data sheet](#) →

Dive into how Darktrace / CLOUD works:

[Read the solution brief](#) →

See Darktrace in action with a personalized meeting:

[Request a demo](#) →

■ About Darktrace

Darktrace is a global leader in AI for cybersecurity that keeps organizations ahead of the changing threat landscape every day. Founded in 2013 in Cambridge, UK, Darktrace provides the essential cybersecurity platform to protect organizations from unknown threats using AI that learns from each business in real-time. Darktrace's platform and services are supported by 2,400+ employees who protect nearly 10,000 customers globally. To learn more, visit <http://www.darktrace.com>.