Docker for Azure Configuration Review

Docker

April 11, 2017 – Version 1.0

Prepared for
Riyaz Faizullabhoy

Prepared by
Andy Schmitz

©2017 – NCC Group

Prepared by NCC Group Security Services, Inc. for Docker. Portions of this document and the templates used in its production are the property of NCC Group and cannot be copied (in full or in part) without NCC Group's permission.

While precautions have been taken in the preparation of this document, NCC Group the publisher, and the author(s) assume no responsibility for errors, omissions, or for damages resulting from the use of the information contained herein. Use of NCC Group's services does not guarantee the security of a system, or that computer intrusions will not occur.

Executive Summary

Synopsis

During February of 2017, Docker engaged NCC Group to conduct a security assessment of the Docker-for-Azure-generated infrastructure. Docker for Azure provides an automatically configured installation of Docker swarm mode on Microsoft's Azure cloud services platform. This assessment was open-ended but was time boxed at ten person-days of effort. Source code was not provided and the scope was limited to the configuration assembled by Docker for Azure.

Scope

NCC Group's evaluation focused on Docker for Azure, which is intended to allow Docker users to deploy Docker swarm mode on Azure both efficiently and securely. The combination of pre-built virtual machines and configuration is intended to provide a securely configured cluster with minimal effort. The configuration built by the Docker for Azure template, as well as basic host hardening on the deployed virtual machines, were in scope.

Features of Docker and Docker swarm mode, such as general container security and Docker-internal network handling, were out of scope, thus this report should not be construed as an assessment of those features.

Testing was performed on the most recent public version of the tools, documented at https://docs.docker.com/docker-for-azure/.

NCC Group's assessment of Docker for Azure included evaluating the access control (Access Control Review on page 4) and network configuration (Network Configuration Review on page 5) provided by Docker in relation to the capabilities offered by Azure and the virtual machines provisioned.

Key Findings

The assessment uncovered a couple of relatively minor network exposure issues:

- Exposure of the Azure instance metadata service to Docker containers.
- Exposure of the Docker swarm mode communication ports to Docker containers.

Limitations

As noted under the scope section, Docker for Azure testing did not include tests of Docker swarm mode functionality. Additionally, this test covered only the standalone installation, not any future integration with Docker Cloud or Docker Datacenter. Finally, no source code or build scripts were available for the virtual machine images used by Docker for Azure, restricting the assessment to the output/results with no review of the input or generation.

Strategic Recommendations

Networking Consider segregating the bridges used by Docker from the rest of the internal network space to avoid attacks on internal resources from containers.

Access Controls and Auditing Docker for Azure currently lacks sufficient forms of access controls and auditing, as there is only a single SSH key provisioned with no built-in mechanisms to add or revoke users, limit user access, or audit user actions. Therefore, NCC Group recommends that enterprise users should consider other Docker products, such as Docker Datacenter, for their production deployments.

Azure Stay current with security features offered by Azure. Azure has added a number of security enhancements since its first release, and future features may help secure Docker for Azure clusters even more. For example, if a feature for intra-virtual network firewalls is introduced, it would provide a better alternative to relying on host-based firewalls for some purposes. It may be useful to provide a mailing list to update Docker for Azure users when new security features are available, as users will be responsible for updating their installations on their own.

Dashboard

Target Metadata

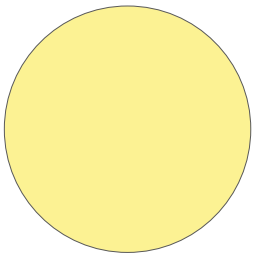
Name	Docker for Azure
Type	Azure Deployment Template
Platforms	Azure
Environment	Private Resource Group on Azure

Engagement Data

Type	Configuration Review
Method	Black-box
Dates	2017-02-06 to 2017-02-17
Consultants	1
Level of effort	10 person-days

Finding Breakdown

Critical Risk issues	0
High Risk issues	0
Medium Risk issues	0
Low Risk issues	3
Informational issues	0
Total issues	3



Category Breakdown

Access Controls	2
Data Exposure	1

Key

Critical High Medium Low Informational

Azure Access Control

Azure provides role-based access control (RBAC) to allow Azure users to restrict the access that other users will have to specific services. Docker for Azure uses only a limited subset of services from Azure (focusing largely on running containers in virtual machines, which are opaque to Azure) making it difficult to use Azure's RBAC to provide meaningful access control for Docker's use of Azure: Docker services do not map easily into Azure services.

Accordingly, Docker for Azure makes no attempt to assign fine-grained policies when creating its resources. An example of this is when creating a Docker for Azure deployment, users must specify an Azure "Resource Group" in which all of the Docker for Azure resources will be created. This Resource Group is then the source of permissions for access to all of the Docker for Azure resources using the recommended deployment template. If any users are set up to have Owner or Contributor access to the Resource Group, they will inherit the permissions for access to view details about the virtual machines, set up load balancer rules, manage storage, or view logs.

While Azure's RBAC does allow users to restrict the inherited access of some users, it is possible that the user who deploys Docker for Azure may not have this ability (as it, too, can be restricted). Therefore, the default configuration of Docker for Azure makes sense: users who already had access to the Resource Group will inherit access to the resources Docker for Azure creates, but no additional access is automatically granted.

Note that users may wish to customize this; for example, by removing direct access to the storage volumes from most developers, or granting log storage location access to auditors or other users. These modifications are outside the scope of this document, but can be made relatively easily from within Azure's portal. Docker for Azure provides sensible names for its resources and most of them are easy to identify.

Docker Access Control

In the existing Docker for Azure implementation, management of the Docker swarm mode is allowed via a single exposed SSH port, secured using a single public key provided at setup. Due to the lack of information about the user's expected source IP addresses, this SSH port is exposed to the entire Internet by default. Once signed in as the `docker` user, users are free to use `sudo` to become root on the Docker management host, but this does not create significant additional attack surface; access to the Docker daemon is effectively equivalent to root access.

NCC Group does have some concerns about groups of operators intending to use the current revision of Docker for Azure to deploy swarms of containers: because all use must go through the `docker` user, the existing configuration makes it difficult to track which users initiated which actions. Additionally, the default configuration requires all administrative users to use the same SSH key, making it difficult to rotate. Organizations wishing to have multiple people administer a single Docker for Azure cluster should strongly consider adding separate SSH keys for each of those administrators to the manager's `authorized_keys` file, rather than sharing one key. This does not provide accountability for individual users, but makes removing access much simpler.

Docker for Azure does not offer, in its current model, any strong form of access controls or auditing and logging mechanisms to Docker nodes, due to the use of a single SSH key outlined above. Docker for Azure should be used in environments where such security requirements are not required. Enterprise users should consider Docker Datacenter and Docker Cloud as they potentially offer more fine-grained access control than Docker for Azure.

Public Network Configuration

At a high level, Docker for Azure's network configuration for exposing Internet access to virtual machines follows best practices for deployment on Azure: virtual machines are deployed on isolated virtual networks and exposed only via explicit load balancers. This ensures that only the expected ports are exposed to the Internet and uses Azure's controls to prevent further access. Rather than a typical "public IP address with a firewall" approach, this configuration is secure by default and exposes only access that is specifically required.

The publicly-accessible management interface is exposed via an SSH port (from the OpenSSH daemon), with public key authentication used and password authentication disabled. While directly exposing SSH to the Internet is a concern in many environments (primarily when password authentication is possible), this configuration provides security comparable to many VPNs: only a private key will grant access and OpenSSH has had no significant publicly-known pre-authentication vulnerabilities for an extended period of time.

Intra-Swarm Network Configuration

Within Docker swarm mode, it is expected that all hosts are able to communicate with one another.[1] Indeed, in Docker for Azure there are no network controls limiting the access between Docker swarm mode worker hosts. Azure does not make such intra-network controls available within a virtual network, so Docker swarm mode would have to rely on host-based firewalls to ensure that no additional ports were exposed. Unfortunately, this provides only limited utility, as an attacker who already has the ability to open ports on a Docker swarm mode worker may well be able to modify the host-based firewall.

In practice, the Docker for Azure use of a single virtual network for a given cluster's workers follows common best practices for Azure network configuration. If Azure later adds support for firewalling individual hosts in a virtual network, Docker for Azure should investigate enabling such a firewall. Docker for Azure should also investigate host-based firewalls as a means to prevent unwanted traffic from reaching Docker swarm mode hosts.

Container Network Configuration

Docker swarm mode makes no effort to constrain the network access from the bridge network given to containers, allowing any container running in the Docker for Azure environment the ability to contact ports used by Docker swarm mode for coordination and monitoring on the Docker swarm mode hosts. This lack of separation has been filed as finding NCC-DOCK_AZURE_2017-002 on page 9. Note that this is not expected to cause significant security issues, as all accessible ports reportedly require authentication with secrets that would not be known to containers. Nonetheless, best practices for security would indicate that the containers do not need to be able to contact the Docker swarm mode hosts.

Similarly, as with many cloud hosts, Azure provides an instance metadata service. In Azure's current implementation of this service, relatively little information is exposed and does not include information that would generally be considered sensitive. This service is available to all containers, giving them information about the host that they are running on. This is a relatively minor leak of information, but could become more concerning if Azure adds additional information to the instance metadata service. This has been noted as finding NCC-DOCK_AZURE_2017-001 on page 8.

Overlay networking used by Docker swarm mode (such as VXLAN) was outside the scope of this assessment, and was not considered as part of the Docker for Azure infrastructure.

[1] https://www.docker.com/products/docker-swarm#/features

Docker for Azure Deployment Best Practices

Docker for Azure follows many industry-standard practices for securing the Azure resources it creates; however, users of Docker for Azure can increase the security of deployed resources in some areas. In particular, NCC Group recommends:

Before setting up Docker for Azure

Use a new Resource Group

Create a new Resource Group for each Docker for Azure deployment, and restrict access to it to only necessary users. In particular, this Resource Group will be fully managed by the Docker swarm mode manager node(s), so a compromise of the Docker environment can lead to a compromise of this Resource Group. Preventing other resources from being included in this Resource Group limits potential exposure in such a scenario. Note that there is generally no reason for users to have access to the Docker for Azure Resource Group, unless they are going to be managing the storage or virtual machines in Azure; even users of the Docker for Azure cluster do not need access to the Resource Group.

After setting up Docker for Azure

Restrict SSH access

If possible, restrict access to the master SSH port via IP address in the Azure portal. (For example, by limiting connections to a corporate VPN source IP address.) Even though only SSH key authentication is enabled in the configuration, limiting the attack surface here can protect against pre-authentication vulnerabilities in OpenSSH.

Restrict resource access

Restrict access to the Docker virtual machine storage accounts (named with random letters and numbers followed by `docker`) and log storage accounts (named with random letters and numbers followed by `logs`) by re-auditing who has access to the Resource Group Docker is running in, removing inherited access from users who do not need access to the storage or logs. For example, your organization may restrict log access to only particular types of users, a policy that can be replicated in your Docker for Azure environment.

Do not share an SSH key

If multiple users will be administering this Docker for Azure installation, consider adding each user's SSH key to the `docker` user's `authorized_keys` file for the Docker manager host, rather than forcing all users to share the same SSH key. Alternatively, look into options such as Docker Cloud or Docker Datacenter (when available) for managing access control to your Docker for Azure cluster.

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Appendix A on page 11.

Title	ID	Risk
Instance Metadata API Accessible to Containers	001	Low
Sensitive Docker Ports Accessible to Containers	002	Low
Service Principal Credentials Shared with Workers	003	Low

Finding	**Instance Metadata API Accessible to Containers**
Risk	**Low** Impact: Low, Exploitability: High
Identifier	NCC-DOCK_AZURE_2017-001
Category	Access Controls
Impact	An attacker with the ability to request files from within a container can gain information about the host the container is running on. The information retrieved is limited to the host instance's name in Azure, its fault domain, its update domain, and information on Azure maintenance that may be scheduled for the host. Note that in other cloud providers,[2] this instance information can also include the custom data provided upon instance initialization, which includes sensitive keys. If Azure implements similar features in the future, it could expose sensitive details to containers.
Description	As with many other cloud providers, Azure provides an "instance metadata" API on a special IP address, 169.254.169.254. This IP address provides an HTTP service that offers some information about the instance, including its name and other metadata. This data is not particularly sensitive, although it can help indicate the number of container hosts deployed, which may be sensitive to some users. Docker containers running on Docker for Azure have full access to the Azure instance metadata API IP address.
Reproduction Steps	1. Set up a Docker for Azure cluster. 2. Log in to the management container. 3. Run a simple container with a shell: `docker run --network=bridge -it ubuntu /bin/bash` (`--network=bridge` is the default, we specify it only to be explicit that this is not the "host" network) 4. Install `curl`: `apt update && apt install -y curl` 5. Run `curl http://169.254.169.254/metadata/v1/maintenance` and note that a response is returned (likely `{}`, indicating no scheduled maintenance). 6. Run `curl http://169.254.169.254/metadata/v1/instanceInfo` and note that a response is returned, indicating the instance hostname, fault domain, and update domain.
Recommendation	Use a host-based firewall on all Docker hosts to prohibit access to `169.254.169.254`. If access is required by the container, implement a push mechanism from the worker node to the container, pushing only the required data.

[2] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html#instancedata-data-retrieval

Finding	**Sensitive Docker Ports Accessible to Containers**
Risk	**Low** Impact: Low, Exploitability: High
Identifier	NCC-DOCK_AZURE_2017-002
Category	Access Controls
Impact	An attacker with access to a container in a Docker for Azure installation can contact the diagnostics, gossip, VXLAN, and Docker swarm mode join ports on the Docker swarm mode hosts.
Description	To perform the work of a standard Docker swarm mode setup, Docker swarm mode hosts expose a number of ports for communicating LAN traffic, cluster join information, and other details. These ports are accessible to any container running in the cluster as no internal firewalling is performed between the cluster hosts and the containers running on them. The Docker team has indicated that the gossip, VXLAN, and Docker swarm mode join ports require authentication details that individual containers would not have. As a result, this is a matter of an increased attack surface being exposed, not sensitive data or network access. The diagnostics port does not expose sensitive data to users, although it can be used to trigger diagnostic reporting, which may be a potential denial of service vector. Docker intends to rate-limit this endpoint in the near future, reducing that risk.
Reproduction Steps	1. Set up a Docker for Azure cluster. 2. Log in to the management container. 3. Run a simple container with a shell: `docker run --network=bridge -it ubuntu /bin/bash` (`--network=bridge` is the default, we specify it only to be explicit that this is not the "host" network) 4. Install `nmap`: `apt update && apt install -y nmap` 5. Run `nmap -p 2377,7946,44554, 10.0.0.4-6` (or different IP addresses, if your cluster manager and workers have different internal IP addresses) and note that ports 7946 (the gossip port) and 44554 (diagnostics) are open on all hosts, and port 2377 (Docker swarm mode join) is open on the swarm manager. 6. Run `nmap -p 4789,1234 -sU 10.0.0.4-6` (or different IP addresses, if your cluster manager and workers have different internal IP addresses) and note that UDP port 4789 (the VXLAN port) is open. (Port 1234 is included as an example of an actually closed port, to differentiate the `open|filtered` status of port 4789.)
Recommendation	Use a host-based firewall on each Docker swarm mode host to prohibit traffic from container bridges to the swarm network.

Finding	**Service Principal Credentials Shared with Workers**
Risk	**Low** Impact: Low, Exploitability: Low
Identifier	NCC-DOCK_AZURE_2017-003
Category	Data Exposure
Location	`customData` variable in https://download.docker.com/azure/stable/Docker.tmpl
Impact	An attacker who compromises a Docker for Azure worker host can gain access to credentials for a service account with full permissions to access the Docker for Azure Resource Group. This can potentially expose the underlying storage of other workers or the manager, widening the scope of the compromise.
Description	Docker for Azure uses a "Service Principal" account intended to allow Docker to manipulate the cluster while it is running. For example, the credentials could be used to create additional worker or manager nodes. These credentials are also used for exposing ports when requested by the user. This Service Principal account's credentials are included in `APP_ID` and `APP_SECRET` environment variables passed to both the manager and worker VMs when instantiated by Docker for Azure. It appears as though manipulation of the Azure environment should be performed only by manager nodes, making it unnecessary to expose the Service Principal credentials to the workers. In particular, if an attacker is able to gain access to a worker host by exploiting a container, they could gain access to these credentials. Restricting the credentials to the manager nodes would limit the effectiveness of such an attack.
Recommendation	Do not give the Service Principal credentials to worker hosts. This likely requires removing them from the `customData` variable in the Docker for Azure template, and only listing them in the custom data for the manager nodes.

Appendix A: Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.