Exploiting Public-Facing Application (version 1.0)

**Cloud Service Label: IaaS, PaaS, SaaS**

Description

The use of software, data, or commands to take advantage of a weakness in an Internet-facing computer system or program in order to cause unintended or unanticipated behavior. The weakness in the system can be a bug, a glitch, or a design vulnerability. These applications are often websites, but can include databases (like SQL), standard services (like SMB or SSH), and any other applications with Internet accessible open sockets, such as web servers and related services.

If an application is hosted on cloud-based infrastructure, then exploiting it may lead to compromise of the underlying instance. This can allow an adversary a path to access the cloud APIs or to take advantage of weak identity and access management policies. It should be assumed that a public web server is a weak link inside a cloud deployment and that some level of compromise is possible. Using Zero Trust principles to isolate the web server to the maximum extent possible inside a cloud deployment will reduce the likelihood of several Cloud tactics and techniques. Docker APIs are commonly exposed publicly and can be used to gain access to a shell on the container, which are also known to run as root by default.

For websites and databases, the OWASP top 10 and CWE top 25 highlight the most common web-based vulnerabilities.

Examples

|  |  |
| --- | --- |
| **Name** | **Description** |
| Shodan.io | Entering queries into Shodan.io like “Product ‘Docker’” or “Product ‘aws’” will yield results to various open APIs and public-facing services. |
| Exposed Docker Port | Identifying an open port associated with Docker (2376 is the most common open port by default) can lead to finding running containers and images. Most containers run with default root privileges, so running *docker -H 192.168.1.7:2376 exec -it <container name> /bin/bash* will automatically give you root once logged in. |

Mitigations

|  |  |  |
| --- | --- | --- |
| **Mitigation** | | **Description** |
| [Application Isolation and Sandboxing](https://attack.mitre.org/mitigations/M1048) | | Application isolation will limit what other processes and system features the exploited target can access. |
| [Exploit Protection](https://attack.mitre.org/mitigations/M1050) | | Web Application Firewalls may be used to limit exposure of applications to prevent exploit traffic from reaching the application. |
| [Network Segmentation](https://attack.mitre.org/mitigations/M1030) | | Segment externally facing servers and services from the rest of the network with a DMZ or on separate hosting infrastructure. |
| Least Privilege | | All access given to users in the cloud environment should be assigned by the necessary privileges needed for team members to complete their job responsibilities. Ensure that temporary access tokens are issued rather than permanent credentials, especially when access is being granted to entities outside of the internal security boundary . |
|  | *AWS* | To implement least privilege in an AWS environment IAM policies will be used. This gives the ability to allow users to perform list, read, write, permissions management, or tagging actions. AWS suggests utilizing *last accessed information* and A*WS CloudTrail event history* to get a better understanding of privileges that might be needed or reduced based on a specific role. Full details can be found at **https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.** |
|  | *Azure* | To implement least privilege in an Azure environment Azure Active Directory roles will be used. Azure outlines different tasks and the least privileged role that are suggested to be associated with the task. Those details can be found at: **https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/roles-delegate-by-task.** To learn how to assign specific roles it can be done via the Azure Active Directory Portal. Instructions on how to assign roles can be found here: **https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/directory-manage-roles-portal.** |
|  | *GCP* | To implement least privilege in GCP it is recommended to use predefined roles (which allow for granular access permissions) instead of primitive roles (roles/owner, roles/editor, and roles/viewer). Full details on the difference between types of roles can be found here: **https://cloud.google.com/iam/docs/understanding-roles.** To assign these roles IAM service accounts are used and complete details can be found at: **https://cloud.google.com/iam/docs/using-iam-securely#least\_privilege.** |
| [Update Software](https://attack.mitre.org/mitigations/M1051) | | Regularly scan externally facing systems for vulnerabilities and establish procedures to rapidly patch systems when critical vulnerabilities are discovered through scanning and through public disclosure. |
| [Vulnerability Scanning](https://attack.mitre.org/mitigations/M1016) | | Regularly scan externally facing systems for vulnerabilities and establish procedures to rapidly patch systems when critical vulnerabilities are discovered through scanning and through public disclosure. |
| Disable Unnecessary Ports and Services | | Ensure ports and services that are publicly accessible are locked down and disabled if necessary. In the case of Docker, port 2376. |
| Assign Permissions to Containers | | By default, many containers will run as root. Before provisioning the container, set explicit permissions and users in either the Dockerfile or User ID at runtime to ensure this does not happen. |

Detection

Monitor application logs for abnormal behavior that may indicate attempted or successful exploitation. Use deep packet inspection to look for artifacts of common exploit traffic, such as SQL injection. Cloud-based Web Application Firewalls tied to proxies may detect improper requests attempting exploitation.

References

1. <https://www.shodan.io/search?query=product+%22docker%22>. Accessed July 14, 2020.

1. <https://medium.com/@riccardo.ancarani94/attacking-docker-exposed-api-3e01ffc3c124>. Accessed July 14, 2020.
2. [Medium. (2020, June). Bryant Hagadorn – Docker and Kubernetes – root vs. Privileged. Retrieved July 14, 2020.](https://itnext.io/docker-and-kubernetes-root-vs-privileged-9d2a37453dec)
3. <https://unit42.paloaltonetworks.com/hunting-the-public-cloud-for-exposed-hosts-and-misconfigurations/>. Accessed July 14, 2020.