Bundesamt
für Sicherheit in der
Informationstechnik

SYS.1.6 Containerisierung

1. Beschreibung

1.1. Einleitung

Der Begriff *Containerisierung* bezeichnet ein Konzept, bei dem Ressourcen eines Betriebssystems partitioniert werden, um Ausführungsumgebungen für Prozesse zu schaffen. Hierbei können je nach verwendetem Betriebssystem unterschiedliche Techniken zum Einsatz kommen, die sich in Funktionsumfang und Sicherheitseigenschaften unterscheiden. Oft wird auch von einer „Betriebssystemvirtualisierung" gesprochen. Es wird jedoch kein vollständiges Betriebssystem virtualisiert, sondern es werden lediglich bestimmte Ressourcen durch einen geteilten Kernel zur Verfügung gestellt. Allgemein wird der Begriff *Container* verwendet, um das entstehende Konstrukt zu bezeichnen.

Bevor leichtfertig komplexe Container-Umgebungen gebaut und verwendet werden, sollte zunächst gründlich abgewogen werden, ob der Aufwand für die Erstellung und den Betrieb einer Container-Umgebungen in einem geeigneten Verhältnis zum tatsächlichen Nutzen steht. Der sachgerechte Betrieb von Container-Umgebungen ist sehr komplex und es sind viele Anforderungen zu beachten. Container können genutzt werden, um als zusätzlicher Trennungsmechanismus eine Härtung der Umgebung zu erreichen, sofern die Art der Containertechnik und die vorgenommene Konfiguration hierfür sachgerecht und geeignet sind.

Es findet eine Unterscheidung zwischen Applikations-Containern (z. B. nach der Spezifikation der Open Container Initiative / OCI) und System-Containern statt. System-Container sind der älteste Typ von Containern, wie z. B. FreeBSD Jails, Solaris Zones, OpenVZ, LXC und LXD. Sie stellen eine Umgebung zur Verfügung, die sich ähnlich einem eigenständigen Betriebssystem verhält, entsprechende Dienste anbieten und mehrere Anwendungen ausführen kann. Applikations-Container hingegen sind speziell für den Fall gedacht, eine einzelne Anwendung auszuführen. Sie folgen dem Lebenszyklus der Anwendung, bieten aber auch innerhalb des Containers keine betriebssystemspezifischen Dienste an. Aus technischer Sicht setzen diese beiden Typen jedoch auf den gleichen Mechanismus zur Trennung, also die Prozessisolation durch den Kernel.

Beim Beispiel Linux-Container (LXC) werden dazu hauptsächlich die Mechanismen *Namespaces* und *cgroups* verwendet, um die Prozessisolation noch zu ergänzen.

- Über Namespaces wird kontrolliert, welche Ressourcen ein Prozess sehen kann. Es gibt sieben unterschiedliche Namespaces: Mount (mnt), Prozess-ID (pid), Netz (net), Interprozess-Kommunikation (ipc), UTS (uts), User ID (user) und Control Group (cgroup).
- Über cgroups wird kontrolliert, welche Ressourcen bzw. welchen Anteil einer Ressource ein Prozess benutzen kann. Zu Ressourcen zählen insbesondere Memory, CPU, BLKIO oder PIDS.

Bei der Verwendung von Namespaces und cgroups sind viele Einzelheiten zu beachten. Unter anderem spielt die Reihenfolge, in der Namespaces geteilt werden, eine entscheidende Rolle. Für diesen Zweck wurden Container-Runtimes entwickelt, wie etwa *runc*, *crun*, *railcar* oder auch *katacontainers*. Die Hauptaufgabe von Container-Runtimes ist das Erstellen einer besonderen Ausführungsumgebung für Prozesse. Sie kommunizieren mit dem Kernel und rufen Syscalls mit den entsprechenden Parametern sowie in der korrekten Reihenfolge auf, um die gewünschte Ausführungsumgebung zu erhalten.

Üblicherweise benötigen Container analog zu einem Betriebssystem ein Dateisystem, in dem die auszuführenden Programme abgelegt werden. Im Container-Umfeld haben sich bestimmte Dateiformate etabliert, um diese Dateisysteme zu beschreiben. Diese werden als *Image,* fälschlicherweise aber teilweise ebenfalls als Container bezeichnet. Abhängig vom verwendeten Container-Typ kann der Inhalt dieser Images von einer einzelnen statisch kompilierten Anwendung bis hin zum vollständigen Inhalt eines Betriebssystems inklusive diverser Ausführungsumgebungen und sonstiger Abhängigkeiten reichen. Images sind transportable, abgeschlossene Einheiten, die in einer Container-Umgebung ausgebracht werden können und alle Komponenten für die Funktionsfähigkeit enthalten.

Neben der Runtime gibt es Container-Engines wie z. B. *Docker*, *Rocket* oder *CRI-O*, die viele Verwaltungsaufgaben übernehmen. In erster Linie bilden sie die Schnittstelle zu Benutzenden und verarbeiten übergebene Befehle. Sie sorgen dafür, dass die benötigten Images zur Verfügung stehen und entsprechende Metadaten vorbereitet sind. Schließlich ruft die Container-Engine die Container-Runtime mit entsprechenden Parametern auf. Die Container-Engine ist damit nicht Teil des Mechanismus Containerisierung, sondern übernimmt hier eine Verwaltungsfunktion.

Weiterhin gibt es unterschiedliche Arten von Containern. Diese unterscheiden sich anhand des Einsatzszenarios und des Lebenszyklus eines Containers. Als *persistenter* Container wird ein Container bezeichnet, der für einen längeren Einsatz gedacht ist. Es kann durchaus valide Gründe geben, in Containern dauerhaft Daten zu speichern. Besonders im Cloud-Umfeld sind jedoch häufig *volatile* Container anzutreffen. Dort haben Container in der Regel eine viel kürzere Lebensdauer, die zudem oft von Orchestrierungswerkzeugen bestimmt wird.

Aus der OCI gibt es Bestrebungen, Standards und Referenzimplementierung zur Verfügung zu stellen. Beispielsweise ist *runc* die Standard-Referenzimplementierung einer Container-Runtime. Andere Container-Runtimes, die kompatibel zum OCI-Standard sind, können somit weitgehend ausgetauscht und verwendet werden.

1.2. Zielsetzung

Ziel dieses Bausteins ist der Schutz von Informationen, die in, von oder mit Containern verarbeitet, angeboten oder darüber übertragen werden. Der Baustein behandelt, wie Container grundsätzlich abgesichert werden können. Dabei wird unterschieden zwischen den Diensten zum Betrieb der Container, also der Software, die für Konfiguration und Verwaltung der Container zuständig ist, sowie den Applikationen und Diensten, die innerhalb der Container ausgeführt werden.

1.3. Abgrenzung und Modellierung

Der Baustein SYS.1.6 *Containerisierung* ist immer anzuwenden, sobald Dienste oder Anwendungen in Containern betrieben werden.

Dieser Baustein betrachtet Container unabhängig von konkreten Produkten. Die Anforderungen orientieren sich an den Fähigkeiten derzeit am Markt verfügbarer Implementierungen. Bei der Produktauswahl ist der Baustein APP.6 *Allgemeine Software* zu berücksichtigen.

Der Baustein ergänzt die Aspekte, die in den Bausteinen SYS.1.1 *Allgemeiner Server* und SYS.1.3 *Server unter Linux und Unix* behandelt werden, um Spezifika von Containerisierung. Die Anforderungen dieser Bausteine sollten vom Host-System erfüllt werden, unabhängig davon, ob dieses auf physischen Servern ausgeführt wird oder virtualisiert ist. Weiterhin gelten die Anforderungen dieser Bausteine

ebenfalls für jede der im Rahmen der Containerisierung erzeugten Userspace-Umgebung. Dabei müssen regelmäßig weitere Bausteine wie z. B. CON.8 *Software-Entwicklung* oder die Bausteine der Teilschicht OPS.1.1 *Kern-IT-Betrieb* berücksichtigt werden. Dies trifft vor allem auf die Erstellung von Images zu.

Typischerweise kommunizieren Container durch virtuelle Netze auf dem Host-System miteinander. Die Bausteine der Teilschichten NET.1 *Netze* und NET.3 *Netzkomponenten* müssen entsprechend berücksichtigt werden.

Sicherheitsanforderungen möglicher Dienste, wie z. B. Webserver (APP.3.2 *Webserver*) oder Webanwendungen (APP.3.1 *Webanwendungen und Webservices*) sind Gegenstand eigener Bausteine, die zusätzlich anzuwenden sind. Sollte das Host-System virtualisiert sein, ist der Baustein SYS.1.5 *Virtualisierung* zu modellieren.

Sollten die Container und die darunterliegende Infrastruktur nicht vollständig selber betrieben und alleinig genutzt werden, sondern werden Teile hiervon durch Dritte bereitgestellt oder von Dritten genutzt, sind zusätzliche Anforderungen der Bausteine OPS.2.3 *Nutzung von Outsourcing* und OPS.2.2 *Cloud-Nutzung* sowie OPS.3.2 *Anbieten von Outsourcing* zu berücksichtigen.

Der Baustein enthält grundsätzliche Anforderungen zur Einrichtung und zum Betrieb von Containerisierung. Die weiteren im Themenumfeld üblichen Dienste, wie z. B. Orchestrierung von Containern, Speichersysteme, virtuelle Netze, Automatisierung für CI/CD-Pipelines oder der Betrieb von Image-Registries, werden hier nicht betrachtet. Ebenso wenig trifft dieser Bausteine Aussagen zu Anforderungen, die für den Bau von Images gelten. Für Anforderungen zur Orchestrierung von Containern mit Kubernetes ist der Baustein APP.4.4 *Kubernetes* zu betrachten.

2. Gefährdungslage

Da IT-Grundschutz-Bausteine nicht auf individuelle Informationsverbünde eingehen können, werden zur Darstellung der Gefährdungslage typische Szenarien zugrunde gelegt. Die folgenden spezifischen Bedrohungen und Schwachstellen sind für den Baustein SYS.1.6 *Containerisierung* von besonderer Bedeutung.

2.1. Schwachstellen oder Schadsoftware in Images

Container werden vorrangig auf Basis von vorgefertigten Images erstellt, die selbst erstellt, aber auch häufig aus dem Internet bezogen werden. Des Weiteren wird Software zunehmend in Form von Images ausgeliefert. Diese Images kann der IT-Betrieb auch für die Erstellung seiner eigenen Images nutzen, indem er Software oder Konfigurationen ergänzt, verändert oder auch entfernt.

Die in den Images enthaltene Software könnte verwundbar und die aus dem Image gestarteten Container könnten dadurch angreifbar sein. Solche Schwachstellen können dem IT-Betrieb auch häufig nicht bekannt sein, da die in den Images enthaltene Software oft nicht in der eigenen Software-Verwaltung erfasst ist. Der IT-Betrieb muss sich grundsätzlich darauf verlassen, dass Updates über den Prozess der Image-Erstellung verfügbar gemacht werden. Von außen betrachtet ist häufig nur aufwendig zu ermitteln, welche Software-Pakete in diesen Images vorhanden sind.

Zudem könnten Images absichtlich integrierte Schadsoftware enthalten, wie z. B. Ransomware oder Kryptominer. Da ein einzelnes Image oft in einer großen Anzahl von Containern ausgebracht wird, kann der resultierende Schaden immens sein.

2.2. Administrative Zugänge ohne Absicherung

Um Container-Dienste auf einem Host zu verwalten, werden administrative Zugänge benötigt. Diese Zugänge sind oft als Dienste realisiert, die entweder lokal oder über das Netz angesprochen werden können. Mechanismen zur Authentisierung und Verschlüsselung der administrativen Zugänge sind häufig vorhanden, aber nicht bei allen Produkten standardmäßig aktiviert.

Wenn Unbefugte auf die Netzsockets oder das Host-System zugreifen können, können sie über ungeschützte administrative Zugänge Befehle ausführen, die zum Verlust der Vertraulichkeit, Integrität und Verfügbarkeit aller auf diesem Host ausgeführten Container führen können.

2.3. Ressourcenkonflikte auf dem Host

Einzelne Container können den Host überlasten und so die Verfügbarkeit aller anderen Container auf dem Host oder auch den Betrieb des Host-Systems selbst gefährden.

2.4. Unberechtigte Kommunikation

Alle Container auf einem Host sind grundsätzlich in der Lage, miteinander, mit dem Host sowie beliebigen anderen Hosts zu kommunizieren. Sofern diese Kommunikation nicht eingeschränkt wird, kann dies ausgenutzt werden, um z. B. andere Container oder Hosts anzugreifen.

Weiterhin besteht die Gefahr, dass Container von außen erreichbar sind, auch wenn dies nicht erwünscht ist. So könnte ein Angriff gegen Dienste, die eigentlich nur intern erreichbar sein sollten, von außen erfolgen. Diese Gefährdung erhöht sich durch die geringere Aufmerksamkeit, die solchen internen Diensten oft entgegengebracht wird. Wird z. B. eine Verwundbarkeit auf einem nur intern eingesetzten Dienst toleriert und ist dieser auch von außen erreichbar, kann dies den gesamten Betrieb erheblich gefährden.

2.5. Ausbruch aus dem Container auf das Host-System

Besteht die Möglichkeit, im Container eigenen Code auszuführen, kann bei einem Angriff möglicherweise die Isolation des Containers gegenüber anderen Containern oder dem Host überwunden und auf andere Container, das Host-System oder die Infrastruktur zugegriffen werden. Es wird auch von einem „Container-Ausbruch" gesprochen. Dieser Angriff kann z. B. über Schwachstellen in Prozessoren, im Betriebssystem-Kernel oder in lokal angebotenen Infrastruktur-Diensten wie z. B. DNS oder SSH erfolgen.

Somit könnte bei einem Angriff die Kontrolle über das Host-System oder andere Systeme aus der Infrastruktur übernommen werden. Es droht der Verlust der Vertraulichkeit, Integrität und Verfügbarkeit aller auf diesem Host ausgeführten Container sowie auf dem Host selbst, falls dort ebenfalls erhöhte Privilegien erlangt werden können.

2.6. Datenverluste durch das Container-Management

Im Rahmen der Verwaltung von Containern können diese ausgeschaltet werden, ohne darin gerade ausgeführter Software die Gelegenheit zu geben, z. B. aktuelle Schreibprozesse abzuschließen (nicht ordnungsgemäßes Herunterfahren). Sollten zu diesen Zeitpunkten Daten durch den Container verarbeitet werden, sind alle diese Daten verloren. Auch Daten, die persistent im Container gespeichert sind, können so dauerhaft verloren gehen.

2.7. Vertraulichkeitsverlust von Zugangsdaten

Die Prinzipien des Aufbaus und der Erstellung von Images für Container setzen voraus, dass Zugangsdaten, z. B. für Datenbanken, im Container verfügbar sind. Über die Images selbst, die Skripte zur Erstellung der Images oder die Versionskontrolle der Skripte können solche Zugangsdaten in unbefugte Hände gelangen.

Oft werden Zugangsdaten auch zum Erstellungszeitpunkt des Containers als Umgebungsvariable verfügbar gemacht. Hier droht ebenfalls der Vertraulichkeitsverlust dieser Daten.

2.8. Ungeordnete Bereitstellung und Verteilung von Images

Im Gegensatz zu klassischen Installationen durch den IT-Betrieb, wo die Kontrolle über die ausgebrachten Anwendungen, Komponenten und Diensten vollständig beim IT-Betrieb selbst liegt, geht diese bei z. B. bei Automatisierung durch CI/CD in Container-Umgebungen verloren. Vielmehr stellt der IT-Betrieb nur eine Plattform bereit, in die Entwickelnde direkt ihre Anwendungen inklusive sämtlicher Abhängigkeiten einbringen und jederzeit verändern können.

3. Anforderungen

Im Folgenden sind die spezifischen Anforderungen des Bausteins SYS.1.6 *Containerisierung* aufgeführt. Der oder die Informationssicherheitsbeauftragte (ISB) ist dafür zuständig, dass alle Anforderungen gemäß dem festgelegten Sicherheitskonzept erfüllt und überprüft werden. Bei strategischen Entscheidungen ist der oder die ISB stets einzubeziehen.

Im IT-Grundschutz-Kompendium sind darüber hinaus weitere Rollen definiert. Sie sollten besetzt werden, insofern dies sinnvoll und angemessen ist.

Zuständigkeiten	Rollen
Grundsätzlich zuständig	IT-Betrieb
Weitere Zuständigkeiten	Keine

Genau eine Rolle sollte *Grundsätzlich zuständig* sein. Darüber hinaus kann es noch *Weitere Zuständigkeiten* geben. Falls eine dieser weiteren Rollen für die Erfüllung einer Anforderung vorrangig zuständig ist, dann wird diese Rolle hinter der Überschrift der Anforderung in eckigen Klammern aufgeführt. Die Verwendung des Singulars oder Plurals sagt nichts darüber aus, wie viele Personen diese Rollen ausfüllen sollen.

3.1. Basis-Anforderungen

Die folgenden Anforderungen MÜSSEN für diesen Baustein vorrangig erfüllt werden.

SYS.1.6.A1 Planung des Container-Einsatzes (B)

Bevor Container eingesetzt werden, MUSS zunächst das Ziel des Container-Einsatzes (z. B. Skalierung, Verfügbarkeit, Wegwerf-Container zur Sicherheit oder CI/CD) festgelegt werden, damit alle sicherheitsrelevanten Aspekte der Installation, des Betriebs und der Außerbetriebnahme geplant werden können. Bei der Planung SOLLTE auch der Betriebsaufwand berücksichtigt werden, der durch Container-Einsatz oder Mischbetrieb entsteht. Die Planung MUSS angemessen dokumentiert werden.

SYS.1.6.A2 Planung der Verwaltung von Containern (B)

Die Verwaltung der Container DARF NUR nach einer geeigneten Planung erfolgen. Diese Planung MUSS den gesamten Lebenszyklus von Inbetrieb- bis Außerbetriebnahme inklusive Betrieb und Updates umfassen. Bei der Planung der Verwaltung MUSS berücksichtigt werden, dass Erstellende eines Containers aufgrund der Auswirkungen auf den Betrieb in Teilen wie Administrierende zu betrachten sind.

Start, Stopp und Überwachung der Container MUSS über die eingesetzte Verwaltungssoftware erfolgen.

SYS.1.6.A3 Sicherer Einsatz containerisierter IT-Systeme (B)

Bei containerisierten IT-Systemen MUSS berücksichtigt werden, wie sich eine Containerisierung auf die betriebenen IT-Systeme und Anwendungen auswirkt, dies betrifft insbesondere die Verwaltung und Eignung der Anwendungen.

Es MUSS anhand des Schutzbedarfs der Anwendungen geprüft werden, ob die Anforderungen an die Isolation und Kapselung der containerisierten IT-Systeme und der virtuellen Netze sowie der

betriebenen Anwendungen hinreichend erfüllt sind. In diese Prüfung SOLLTEN die betriebssystemeigenen Mechanismen mit einbezogen werden. Für die virtuellen Netze nimmt der Host die Funktion einer Netzkomponente wahr, die Bausteine der Teilschichten NET.1 *Netze* und NET.3 *Netzkomponenten* MÜSSEN entsprechend berücksichtigt werden. Logische und Overlay-Netze MÜSSEN ebenfalls betrachtet und modelliert werden. Weiterhin MÜSSEN die eingesetzten containerisierten IT-Systeme den Anforderungen an die Verfügbarkeit und den Datendurchsatz genügen.

Im laufenden Betrieb SOLLTEN die Performance und der Zustand der containerisierten IT-Systeme überwacht werden (sogenannte Health Checks).

SYS.1.6.A4 Planung der Bereitstellung und Verteilung von Images (B)

Der Prozess zur Bereitstellung und Verteilung von Images MUSS geplant und angemessen dokumentiert werden.

SYS.1.6.A5 Separierung der Administrations- und Zugangsnetze bei Containern (B)

Die Netze für die Administration des Hosts, die Administration der Container und deren Zugangsnetze MÜSSEN dem Schutzbedarf angemessen separiert werden. Grundsätzlich SOLLTE mindestens die Administration des Hosts nur aus dem Administrationsnetz möglich sein.

Es SOLLTEN nur die für den Betrieb notwendigen Kommunikationsbeziehungen erlaubt werden.

SYS.1.6.A6 Verwendung sicherer Images (B)

Es MUSS sichergestellt sein, dass sämtliche verwendeten Images nur aus vertrauenswürdigen Quellen stammen. Es MUSS eindeutig identifizierbar sein, wer das Image erstellt hat.

Die Quelle MUSS danach ausgewählt werden, dass die im Image enthaltene Software regelmäßig auf Sicherheitsprobleme geprüft wird und diese behoben sowie dokumentiert werden. Die Quelle MUSS diesen Umgang mit Sicherheitsproblemen zusichern und einhalten.

Die verwendete Version von Basis-Images DARF NICHT abgekündigt („deprecated") sein. Es MÜSSEN eindeutige Versionsnummern angegeben sein. Wenn ein Image mit einer neueren Versionsnummer verfügbar ist, MUSS im Rahmen des Patch- und Änderungsmanagement geprüft werden, ob und wie dieses ausgerollt werden kann.

SYS.1.6.A7 Persistenz von Protokollierungsdaten der Container (B)

Die Speicherung der Protokollierungsdaten der Container MUSS außerhalb des Containers, mindestens auf dem Container-Host, erfolgen.

SYS.1.6.A8 Sichere Speicherung von Zugangsdaten bei Containern (B)

Zugangsdaten MÜSSEN so gespeichert und verwaltet werden, dass nur berechtigte Personen und Container darauf zugreifen können. Insbesondere MUSS sichergestellt sein, dass Zugangsdaten nur an besonders geschützten Orten und nicht in den Images liegen. Die von der Verwaltungssoftware des Container-Dienstes bereitgestellten Verwaltungsmechanismen für Zugangsdaten SOLLTEN eingesetzt werden.

Mindestens die folgenden Zugangsdaten MÜSSEN sicher gespeichert werden:

- Passwörter jeglicher Accounts,
- API-Keys für von der Anwendung genutzte Dienste,
- Schlüssel für symmetrische Verschlüsselungen sowie
- private Schlüssel bei Public-Key-Authentisierung.

3.2. Standard-Anforderungen

Gemeinsam mit den Basis-Anforderungen entsprechen die folgenden Anforderungen dem Stand der Technik für diesen Baustein. Sie SOLLTEN grundsätzlich erfüllt werden.

SYS.1.6.A9 Eignung für Container-Betrieb (S)

Die Anwendung oder der Dienst, die oder der im Container betrieben werden soll, SOLLTE für den Container-Betrieb geeignet sein. Dabei SOLLTE berücksichtigt werden, dass Container häufiger für die darin ausgeführte Anwendung unvorhergesehen beendet werden können. Die Ergebnisse der Prüfung nach SYS.1.6.A3 *Sicherer Einsatz containerisierter IT-Systeme* SOLLTE nachvollziehbar dokumentiert werden.

SYS.1.6.A10 Richtlinie für Images und Container-Betrieb (S)

Es SOLLTE eine Richtlinie erstellt und angewendet werden, die die Anforderungen an den Betrieb der Container und die erlaubten Images festlegt. Die Richtlinie SOLLTE auch Anforderungen an den Betrieb und die Bereitstellung der Images enthalten.

SYS.1.6.A11 Nur ein Dienst pro Container (S)

Jeder Container SOLLTE jeweils nur einen Dienst bereitstellen.

SYS.1.6.A12 Verteilung sicherer Images (S)

Es SOLLTE angemessen dokumentiert werden, welche Quellen für Images als vertrauenswürdig klassifiziert wurden und warum. Zusätzlich SOLLTE der Prozess angemessen dokumentiert werden, wie Images bzw. die im Image enthaltenen Softwarebestandteile aus vertrauenswürdigen Quellen bezogen und schließlich für den produktiven Betrieb bereitgestellt werden.

Die verwendeten Images SOLLTEN über Metadaten verfügen, die die Funktion und die Historie des Images nachvollziehbar machen. Digitale Signaturen SOLLTEN jedes Image gegen Veränderung absichern.

SYS.1.6.A13 Freigabe von Images (S)

Alle Images für den produktiven Betrieb SOLLTEN wie Softwareprodukte einen Test- und Freigabeprozess gemäß dem Baustein OPS.1.1.6 *Software-Test und Freigaben* durchlaufen.

SYS.1.6.A14 Aktualisierung von Images (S)

Bei der Erstellung des Konzeptes für das Patch- und Änderungsmanagement gemäß OPS.1.1.3 *Patch- und Änderungsmanagement* SOLLTE entschieden werden, wann und wie die Updates der Images bzw. der betriebenen Software oder des betriebenen Dienstes ausgerollt werden. Bei persistenten Containern SOLLTE geprüft werden, ob in Ausnahmefällen ein Update des jeweiligen Containers geeigneter ist, als den Container vollständig neu zu provisionieren.

SYS.1.6.A15 Limitierung der Ressourcen pro Container (S)

Für jeden Container SOLLTEN Ressourcen auf dem Host-System, wie CPU, flüchtiger und persistenter Speicher sowie Netzbandbreite, angemessen reserviert und limitiert werden. Es SOLLTE definiert und dokumentiert sein, wie das System im Fall einer Überschreitung dieser Limitierungen reagiert.

SYS.1.6.A16 Administrativer Fernzugriff auf Container (S)

Administrative Zugriffe von einem Container auf den Container-Host und umgekehrt SOLLTEN prinzipiell wie administrative Fernzugriffe betrachtet werden. Aus einem Container SOLLTEN KEINE administrativen Fernzugriffe auf den Container-Host erfolgen. Applikations-Container SOLLTEN keine Fernwartungszugänge enthalten. Administrative Zugriffe auf Applikations-Container SOLLTEN immer über die Container-Runtime erfolgen.

SYS.1.6.A17 Ausführung von Containern ohne Privilegien (S)

Die Container-Runtime und alle instanziierten Container SOLLTEN nur von einem nicht-privilegierten System-Account ausgeführt werden, der über keine erweiterten Rechte für den Container-Dienst und das Betriebssystem des Host-Systems verfügt oder diese Rechte erlangen kann. Die Container-Runtime SOLLTE durch zusätzliche Maßnahmen gekapselt werden, etwa durch Verwendung der Virtualisierungserweiterungen von CPUs.

Sofern Container ausnahmsweise Aufgaben des Host-Systems übernehmen sollen, SOLLTEN die Privilegien auf dem Host-System auf das erforderliche Minimum begrenzt werden. Ausnahmen SOLLTEN angemessen dokumentiert werden.

SYS.1.6.A18 Accounts der Anwendungsdienste (S)

Die System-Accounts innerhalb eines Containers SOLLTEN keine Berechtigungen auf dem Host-System haben. Wo aus betrieblichen Gründen diese Berechtigung notwendig ist, SOLLTE diese nur für unbedingt notwendige Daten und Systemzugriffe gelten. Der Account im Container, der für diesen Datenaustausch notwendig ist, SOLLTE im Host-System bekannt sein.

SYS.1.6.A19 Einbinden von Datenspeichern in Container (S)

Die Container SOLLTEN NUR auf die für den Betrieb notwendigen Massenspeicher und Verzeichnisse zugreifen können. Nur wenn Berechtigungen benötigt werden, SOLLTEN diese explizit vergeben werden. Sofern die Container-Runtime für einen Container lokalen Speicher einbindet, SOLLTEN die Zugriffsrechte im Dateisystem auf den Service-Account des Containers eingeschränkt sein. Werden Netzspeicher verwendet, so SOLLTEN die Berechtigungen auf dem Netzspeicher selbst gesetzt werden.

SYS.1.6.A20 Absicherung von Konfigurationsdaten (S)

Die Beschreibung der Container-Konfigurationsdaten SOLLTE versioniert erfolgen. Änderungen SOLLTEN nachvollziehbar dokumentiert sein.

3.3. Anforderungen bei erhöhtem Schutzbedarf

Im Folgenden sind für diesen Baustein exemplarische Vorschläge für Anforderungen aufgeführt, die über dasjenige Schutzniveau hinausgehen, das dem Stand der Technik entspricht. Die Vorschläge SOLLTEN bei erhöhtem Schutzbedarf in Betracht gezogen werden. Die konkrete Festlegung erfolgt im Rahmen einer individuellen Risikoanalyse.

SYS.1.6.A21 Erweiterte Sicherheitsrichtlinien (H)

Erweiterte Richtlinien SOLLTEN die Berechtigungen der Container einschränken. Mandatory Access Control (MAC) oder eine vergleichbare Technik SOLLTE diese Richtlinien erzwingen. Die Richtlinien SOLLTEN mindestens folgende Zugriffe einschränken:

- eingehende und ausgehende Netzverbindungen,
- Dateisystem-Zugriffe und
- Kernel-Anfragen (Syscalls).

Die Runtime SOLLTE die Container so starten, dass der Kernel des Host-Systems alle nicht von der Richtlinie erlaubten Aktivitäten der Container verhindert (z. B. durch die Einrichtung lokaler Paketfilter oder durch Entzug von Berechtigungen) oder zumindest Verstöße geeignet meldet.

SYS.1.6.A22 Vorsorge für Untersuchungen (H)

Um Container im Bedarfsfall für eine spätere Untersuchung verfügbar zu haben, SOLLTE ein Abbild des Zustands nach festgelegten Regeln erstellt werden.

SYS.1.6.A23 Unveränderlichkeit der Container (H)

Container SOLLTEN ihr Dateisystem während der Laufzeit nicht verändern können. Dateisysteme SOLLTEN nicht mit Schreibrechten eingebunden sein.

SYS.1.6.A24 Hostbasierte Angriffserkennung (H)

Das Verhalten der Container und der darin betriebenen Anwendungen und Dienste SOLLTE überwacht werden. Abweichungen von einem normalen Verhalten SOLLTEN bemerkt und gemeldet werden. Die Meldungen SOLLTEN im zentralen Prozess zur Behandlung von Sicherheitsvorfällen angemessen behandelt werden.

Das zu überwachende Verhalten SOLLTE mindestens umfassen:

- Netzverbindungen,
- erstellte Prozesse,
- Dateisystem-Zugriffe und
- Kernel-Anfragen (Syscalls).

SYS.1.6.A25 Hochverfügbarkeit von containerisierten Anwendungen (H)

Bei hohen Verfügbarkeitsanforderungen der containerisierten Anwendungen SOLLTE entschieden werden, auf welcher Ebene die Verfügbarkeit realisiert werden soll (z. B. redundant auf der Ebene des Hosts).

SYS.1.6.A26 Weitergehende Isolation und Kapselung von Containern (H)

Wird eine weitergehende Isolation und Kapselung von Containern benötigt, dann SOLLTEN folgende Maßnahmen nach steigender Wirksamkeit geprüft werden:

- feste Zuordnung von Containern zu Container-Hosts,
- Ausführung der einzelnen Container und/oder des Container-Hosts mit Hypervisoren,
- feste Zuordnung eines einzelnen Containers zu einem einzelnen Container-Host.

4. Weiterführende Informationen

4.1. Wissenswertes

Weiterführende Informationen zu Gefährdungen und Sicherheitsmaßnahmen im Bereich Container finden sich unter anderem in folgenden Veröffentlichungen:

- NIST 800-190
  https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-190.pdf
- CIS Benchmark Docker
  https://www.cisecurity.org/benchmark/docker/
- OCI - Open Container Initiative
  https://www.opencontainers.org/
- CNCF - Cloud Native Computing Foundation
  https://www.cncf.io/
- SANS Checkliste
  https://www.sans.org/reading-room/whitepapers/auditing/checklist-audit-docker-containers-37437
- Docker Security Guide
  https://docs.docker.com/engine/security/