容器安全成熟度验证标准

本报告针对以 Docker 为代表的容器解决方案，建立一个安全要求和控制框架，主要是解决在规范设计、开发和测试的过程中所需的功能性和非功能性安全控制。该框架有 2 个核心作用：

- 帮助组织机构的开发和运维确保容器的安全性。
- 帮助容器安全产品和服务供应商以及最终用户调整他们的产品和具体要求。

容器安全成熟度标准

容器安全成熟度可以分为 3 个层级，相关要求逐渐提升。

- Level1（基础级安全）：适用于所有容器项目。
- Level2（标准级安全）：适用于那些有额外安全保护需求的容器项目，例如需要处理敏感数据或者业务逻辑。
- Level3（高级安全）：适用于那些执行高价值交易、包含个人敏感数据的最关键容器项目，或任何需要最高信任级别的容器。

不同层级的容器安全成熟度有不同要求列表（Checklist）。这些检查要求都可以映射到具体的容器安全产品功能或能力上去，而这些能力是必须内置到容器中去的。

对于这个标准最佳实践是将其作为“挂图作战”看板，基于该标准创建适用于具体项目、平台、组织的安全检查列表（Checklist）。根据自身企业情况，以及结合该容器安全标准成熟度框架可以增强组织对于容器安全的认知。

检测标准：12 个维度，100+Checklist

场景 1：组织层面

除了技术控制之外，容器安全检查标准还要求流程到位，以确保在规划架构时就已明确要解决的安全问题，并且所有组件的功能和安全规则都是已知的。

场景 1，从组织层面列出了与容器解决方案的流程、架构和设计相关的要求。因此，此类别无法映射到技术测试用例，而必须在流程级别上进行处理。但是，因每个组织的结构不同，并且每个设置步骤都有不同的组织和流程要求，文本在组织层面控制要求，仅涵盖了一些基本方面。

描述	L1	L2	L3
1.检查开发人员，尤其是负责 DevOps 的人员是否接受了相关的定期安全培训。	√	√	√
2.检查管理人员是否接受项目中有关的安全方面的定期培训。			√
3.检查所有处理的数据是否已经按照内部数据分类标准进行分类。	√	√	√
4.检查每个应用程序都能提供其安全需求以及它们将如何解决的相关信息。		√	√
5.检查已知安全风险和漏洞是否已按预设的管理流程迅速消除并集中管理。		√	√
6.检查有关容器基础设施的规则和职责是否已明确，例如镜像的准入准出。		√	√

场景 2：基础设施

不同行业的底层基础设施安全设置可能有所不同，但基础设施必须为上层提供所需的安全性。确保经过安全验证的容器解决方案能够满足以下要求：

- 确保基础设施能够提供足够的资源。
- 加固包括容器平台在内的基础架构。

描述	L1	L2	L3
1.检查容器整体架构和设计，包括内部和外部网络。	√	√	√
2.检查基础设施，包括其所有组件（节点、网络、容器等）是否已记录在案。	√	√	√
3.检查所使用组件都得到维护并相互兼容（OS、引擎、UCP、DTR 等）。	√	√	√
4.检查是否为所有节点分配足够的资源以使其稳定运行。	√	√	√
5.检查容器可用资源是否设定有限制。		√	√
6.验证 SELinux 或 AppArmor 是否已启用并在所有节点上运行。			√
7.检查节点的 Docker 引擎是否定期自动更新和应用是最新版的。	√	√	√
8.检查是否使用了允许回滚的灰度部署/发布策略。		√	√
9.检查容器的配置是否支持实时还原可用。		√	√
10.检查 Docker 的配置权限是否仅限于用户实际所需的访问。	√	√	√
11.验证所有节点（包括操作系统）是否都定期经过自动安全扫描。		√	√
12.检查所有节点是否都使用了容器专用操作系统来代替通用操作系统。			√
13.检查所有节点是否都使用了常规的加固操作。	√	√	√
14.检查是否使用了 Docker 默认配置值（除非特殊申明）。	√	√	√
15.检查是否已经尽可能限制 node 远程访问，包括 SSH 或 RDP 等。	√	√	√

场景 3：容器层面

容器解决方案核心部分就是应用容器。容器本身不仅仅包含了服务和应用逻辑，还包括与其它系统或容器的交互，这个过程涉及诸多需要保护的敏感数据。确保经过检查验证容器安全方案能够满足以下要求：

- 确保容器以最小权限运行。
- 加固容器内的服务并最小化攻击面。
- 利用容器技术自身的安全特性。

描述	L1	L2	L3
1.检查容器 root 权限（初始化过程除外，但在完成后应删除 root 权限）。		√	√
2.检查是否启用了用户命名空间。		√	√
3.检查是否在每个镜像中创建了新用户，并使用该用户执行容器内所有操作。			√
4.检查那些有特定 seccomp-profile 配置文件，是否已应用于每个容器。			√
5.检查容器在运行过程中是否被授予了其它额外权限。	√	√	√
6.通过哈希值验证所有基础镜像，而不是基于名称或者标签。			√
7.检查在进入生产环境之前，每个镜像签名是否已经被验证。			√
8.检查镜像是否只安装了必须软件包。	√	√	√
9.检查根文件系统是否设置为可读模式进行挂载。		√	√
10. 检查那些特殊容器(例为了故障排除)，是否已经删除并已替换为新容器。		√	√
11. 验证 Dockerfiles 是否使用 COPY 指令而不是 ADD 指令，除非来源是可信。	√	√	√
12. 检查远程登录管理服务是否已禁用，包括但不限于 SSH、RDP 等。	√	√	√
13. 检查公开服务（例如 etcd）是否仅限于那些可信系统或经过身份验证系统。	√	√	√
14. 使用--pids-limit 检查容器内允许进程数是否已限制为确定数值。			√
15. 检查容器未安装任何 Docker socket，除非它们用于监视或管理。如果需要 Docker socket，检查是否设置为只读，并相应地限制容器的访问。	√	√	√

场景 4：编排管理

当容器数量达到一定程度时，就需要一个编排管理工具。编排管理工具可以帮助管理、跟踪正在发生的事情。由于编排工具是容器基础设施中不可缺少的一部分，因此编排工具的安全级别将直接影响基础设施的所有其他方面。确保经过验证的容器安全解决方案能够满足以下安全要求：

- 确保编排工具的正常运行时间。
- 对编排工具进行加固。
- 能够实现与编排工具自动化交互，避免人为错误。

描述	L1	L2	L3
1.检查编排管理节点是否已设置冗余，并能支持高可用性要求。	√	√	√
2.检查部署管理器节点数是否为奇数，并且至少大于 3。	√	√	√
3.验证管理节点是否分布在多个数据中心和可用区。		√	√
4.检查管理节点是否会在启动自动锁定情况下运行。			√
5.检查编排工具是否定期平衡活跃容器。.	√	√	√
6.检查管理器节点是否承担工作任务和运行容器。	√	√	√
7.检查验证预定义的标签是否能够正确识别和管理所有资源。		√	√
8.检查是否已经删除那些不需要的容器。		√	√
9.检查同一个节点上运行的容器都拥有同样数据分类级别。			√
10.验证同一个节点上运行的容器具有相同暴露面（比如面向互联网）。			√

场景 5：镜像分发

容器运行的前提是构建镜像。镜像定义了容器运行的内容，包括软件包、版本号等。每个容器的镜像安全对于确保容器安全运行至关重要。一个安全的容器解决方案至少需要满足以下几个点：

- 加固镜像。
- 镜像内部没有存储敏感数据。
- 检查镜像是否存在易受攻击的组件。

描述	L1	L2	L3
1.检查镜像仓库数是否为奇数，并且至少大于 3。			√
2.检查镜像仓库是否启用了垃圾收集并能定期运行。	√	√	√
3.验证所有镜像是否经过定期的自动化安全扫描。		√	√
4.检查验证容器是否始终是基于最新的镜像生成而不是本地缓存镜像生成。	√	√	√
5.检查所有镜像是否都有标签，并只允许生产/Master 节点使用默认最新标签。		√	√

场景 6：Secrets 管理

生产系统通常包括诸多 secrets 资源，包括加密秘钥、用户名密码等。本节主要说明如何处理此类敏感信息，确保经过验证容器安全解决方案能够满足以下条件：

- 保护敏感信息。
- 验证是否对加密信息进行安全处理。
- 定期轮换加密秘钥。

描述	L1	L2	L3
1.检查是否使用了 RBAC 模型来管理访问控制。		√	√
2.检查 Docker Content Trust 是否已启动并强制执行。			√
3.检查是否使用 Docker Secrets 来处理 API 秘钥、密码等敏感信息，确认没有将敏感信息存储在 Dockerfile 或者 Docker-Compose 中情况。	√	√	√
4.检查加入编排工具的密钥是否定期轮换。		√	√
5.验证如果启动自动锁定，检查自动锁定密钥是否定期轮换。		√	√
6.检查 node 节点证书是否定期更换。		√	√
7.检查 CA 证书是否定期更换。		√	√
8.检查用于生成内部集群间通信的相互 TLS 身份验证的 CA 证书。		√	√
9.检查使用的 SSL/TLS 证书是否经过验证。	√	√	√
10.检查是否使用了 Secrets 管理安全解决方案处理 Secrets 对象。		√	√

场景 7：网络安全

当下所有应用和服务都不可能单一独立的，通常都是通过网络连接进行交互，形成一个整体的系统服务。网络安全是一个单独安全门类，但是容器技术会影响网络安全，那么在使用容

器过程中，可以在哪些方面提高网络安全呢？

经过验证合格的容器安全解决方案应该满足以下安全需求：

- 选择好的网络驱动程序并进行正确配置。
- 禁用不需要的功能和应用限制。
- 网络传输的数据需要经过加密。

描述	L1	L2	L3
1.检查是否使用了准备就绪网络驱动程序。	√	√	√
2.检查负载均衡功能是否已经激活。		√	√
3.检查 Docker userland 代理是否已禁用。		√	√
4.检查是否未使用默认网桥。	√	√	√
5.检查 Docker 配置是否为默认配置，确认不同容器之间无法进行网络通信。		√	√
6.检查 docker 是否存在允许修改 iptable 规则情况。	√	√	√
7.检查已发布的端口是否已经分配给对应节点的网络接口。	√	√	√
8.检查管理和应用流量是否有不同网络接口进行区分。			√
9.检查是否为每个应用程序分配了至少一个单独、隔离的应用层网络，以确保 L3 级别隔离。		√	√
10.检查容器或者 node 节点之间网络连接是否已经加密。		√	√
11.检查使用的子网是否与其它子网会重叠。	√	√	√
12.检查发布的端口数量是否限制在必要的最小值。	√	√	√

场景 8：存储安全

因为容器的生命周期是非常短暂的，因此为容器提供安全可控的存储能力非常重要，确保数据的可用性、完整性和访问控制措施非常重要。确保经过验证的容器安全解决方案至少需要满足以下要求：

- 选择高质量存储驱动并正确配置。
- 确保数据存储在 node 节点持久化。

描述	L1	L2	L3
1.检查是否使用了准备就绪的存储。	√	√	√
2.检查镜像仓库是否有冗余并且位于安全的网络区域中。	√	√	√
3.检查是否使用了合格且经过安全测试的数据存储驱动，确保应用数据的可复制和可用性。	√	√	√
4.确认数据未直接存储在容器内，而是存储在相应的数据卷或者挂载点上。	√	√	√
5.检查是否有定期进行备份持久数据，并测试恢复能力。	√	√	√

场景 9：日志和监控

日志记录和监控可以确保在出现问题之后进行响应分析，确保经过验证的容器安全解决方案

需要满足以下需求：

- 需要有集中日志记录和监控能力。
- 监控所有组件。

描述	L1	L2	L3
1.检查是否已经对底层操作系统、Docker 引擎、容器、进程进行日志记录。		√	√
2.检查是否对节点和容器级别使用的资源进行监控。		√	√
3.检查存储端是否进行监控。		√	√
4.检查是否基于容器监控检查功能对所有容器及他们状态进行监控。		√	√
5.检查确认所有日志是否都已传输并存储到一个中心位置。		√	√
6.检查在生产环境中，docker 的日志级别设置为 info	√	√	√

场景 10：安全集成

容器安全解决方案通常都不是独立的，需要与各种不同系统集成，包括 IAM、CI/CD 等。任何交互过程都会对容器产生潜在威胁，反之亦然。确保经过审核解决方案满足以下要求：

- 能够将安全能力集成到现有的基础设施中。
- 利用现有网络基础设施。

描述	L1	L2	L3
1.检查编排工具和仓库（DTR 等）是否已集成到现有基础架构。		√	√
2.检查 CI/CD 工具和系统是否已经连接到 Docker 基础架构，可实现自动化对节点、镜像、网络的更改、测试和部署。		√	√
3.检查是否可以自动设置其它节点，并以与现在节点相同方式进行配置。		√	√
4.检查是否实施集中日志管控系统，能够跟踪到容器及组件所有变更。		√	√
5.检查内外部是否使用了诸如 Consul、Zookeeper、Eureka、Etcd 甚至是 DNS 之类的发现和注册服务。		√	√
6.检查用户和角色是否已经映射到现有 IAM 方案中。			√

场景 11：灾难恢复

当出现安全问题之后，最重要是否能按照预期的灾难恢复工作，并且确保停机时间比较短才行。确保经过验证的容器安全解决方案能够满足以下基本要求：

- 定期创建备份。
- 自动化恢复。
- 利用自愈能力。

描述	L1	L2	L3
1.检查是否执行了定期备份（UCP、DTR、Swarm），每周至少执行一次。	√	√	√
2.检查基础设施是否支持自动恢复，能够定期测试。	√	√	√

3.检查基础设施和容器引擎升级、降级是否支持自动化、可记录、定期测试。			√
4.检查单体应用、服务是否支持自动恢复、可记录、定期测试。		√	√
5.检查是否为每个容器启用了故障重启的策略。		√	√

场景 12：测试

技术是快速迭代发展，因此容器安全保护并不是一次性工作，而应该定期进行检查和验证。经过验证的容器安全解决方案应该满足以下要求：

- 从故障中恢复。
- 确保安全设置真实有效。
- 容器安全解决方案当前状态相关文档。

描述	L1	L2	L3
1.检查所有用户和组所拥有的资源权限是否符合规范和文档要求。	√	√	√
2.检查应用程序、容器资源显示是否符合预定义的工作要求。		√	√
3.检查每个服务是否都支持完全自动化的方式重新创建。		√	√
4.检查证书和密钥是否按照规范定期更换。			√
5.检查所有服务的配置、镜像和网络是否支持滚动更新或回滚降级。		√	√
6.检查节点和 Docker 引擎是否为最新。	√	√	√
7.检查负载均衡策略是否按照预定设置在工作。		√	√
8.检查所有服务是否都可以从节点或单个容器的故障中恢复。		√	√
9.检查是否可以在完全失败情况下为所有服务恢复备份。	√	√	√
10.检查是否定期运行并且通过了 Docker Security Bench（CIS）。		√	√

如何使用这个成熟度标准

不同的威胁有不同的动机。一些行业具有独特的信息和技术资产，以及需要满足特定行业领域的法律法规。虽然，不同行业监管要求不一样，会存在一些独特的标准以及相关威胁也存在一定的差异。但是不管是哪个行业，机会主义攻击者都会寻找那些容易被利用的漏洞，这就是为什么所有基于容器的项目都建议需要满足 Level1 的安全管控要求 。

此外，也建议组织根据其所在单位业务的差异性，深入研究那些行业所特有的风险特征。Level3 级别的要求，只适用于那些一旦发生意外可能会严重影响组织正常运行的情况。

对于容器安全成熟度标准，针对不同角色有不同使用场景，例如，作为具体安全架构指南、作为检查清单、作为安全培训等等。

- **作为详细的安全架构指南**

容器安全成熟度标准常见的用途之一是作为安全架构师的参考框架。两大安全架构框架 SABSA（Sherwood Applied Business Security Architecture）和 TOGAF（The Open Group Application Framework） 缺少完成应用程序和容器安全架构审查所需的大量信息。容器安全成熟度验证标准可用于通过允许安全架构师为常见问题选择更好的控制来填补这些空白。

- **作为新的检测清单**

组织机构可以将容器安全成熟度标准作为检测清单，基于自身企业安全要求选择三个级别中的一个，或者综合三个级别的检测清单，形成自己的 checklist。

- **安全培训**

容器安全成熟度标准是来源于实战经验的总结，因此有很强的实践意义，可以作为安全培训的材料。