计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(8):247−251 [doi: 10.15888/j.cnki.csa.005893]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

Docker Remote API 未授权访问漏洞利用工具①

孙　建[1], 蔡　翔[1], 王　潇[1], 夏雨潇[2]

[1](国网安徽省电力公司 电力科学研究院, 合肥 230601)
[2](国网安徽省电力公司 安庆供电公司, 安庆 246003)

摘　要: 通过分析 Docker Remote API 未授权访问漏洞的原理, 设计并实现 Docker Remote API 未授权访问漏洞利用工具. 该工具弥补了当前网络环境下此种漏洞检测工具的缺失, 并提供批量检测功能, 大大提高了漏洞检测效率, 为漏洞修复等安全工作打下基础, 是一种效果好成本低的 Web 应用安全防护方案.

关键词: Docker Remote API; 漏洞利用; 漏洞检测; Web 应用安全

引用格式: 孙建,蔡翔,王潇,夏雨潇.Docker Remote API 未授权访问漏洞利用工具.计算机系统应用,2017,26(8):247–251. http://www.c-s-a.org.cn/1003-3254/5893.html

Docker Remote API Unauthorized Access Vulnerability Exploitation Tool

SUN Jian[1], CAI Xiang[1], WANG Xiao[1], XIA Yu-Xiao[2]

[1](State Grid Anhui Electric Power Research Insitute, Hefei 230601, China)
[2](State Grid Anqing Electric Power Suply, Anqing 246003, China)

Abstract: Through the analysis of Docker Remote API Unauthorized Access Vulnerability, this paper proposes a vulnerability exploitation tool. The tool fills the gap of vulnerability detection tools and provides the batch testing function. It achieves high efficiency of vulnerability detection and lays the foundation for web security work such as bug fixes. It is effective and has low cost for the improvement of web security.

Key words: Docker Remote API; vulnerability exploitation; vulnerability detection; Web application security

随着信息技术的快速发展, 互联网网络安全问题也越发严重, 2016 Internet Security Threat Report 表明 2015 年 0-day 漏洞的数量达到 54 个, 相比于 2014 年增长了 125%; 5 亿人个人信息被盗或丢失; 75% 的网站都存在漏洞[1]. 各种 Web 应用的安全事件严重影响了 Web 应用的发展, 其中未授权访问漏洞的应用攻击尤为严重. MongoDB 数据库未授权访问漏洞导致攻击者可任意查看数据库中的数据; Memcache 未授权访问漏洞不仅会导致 Memcached 中数据可被直接读取泄漏和恶意修改, 而且从 Memcached 中读取的数据则更容易被开发者认为是可信的, 或者是已经通过安全校验的, 因此更容易导致安全问题; Redis 因配置不当可以未授权访问, 被攻击者恶意利用, 如果 Redis 以 root 身份运行, 黑客可以给 root 账户写入 SSH 公钥文件, 直接通过 SSH 登录受害服务器. 2016 年 5 月, 一些关于 Docker Remote API 未授权访问导致代码泄露、获取服务器 root 权限的漏洞被提交, 造成了严重的影响, 多家大型网站可以通过此漏洞获取全站代码和服务器 root 权限. 漏洞虽然引起了足够的重视, 但是漏洞公布的细节着眼于漏洞原理分析, 而没有很成型的概念证明(POC)发布, 而且就漏洞防护措施而言, 使用 Web 应用程序漏洞检测工具自动生成 POC 代码并进行渗透测试[2]是最有效简便的检测此漏洞的方式.

因此, 本文在对其漏洞原理进行全面分析的基础上, 充分考虑针对此漏洞的检测和利用方案, 设计并实现了 Docker Remote API 未授权访问漏洞利用工具, 为

① 收稿时间: 2016-11-21; 采用时间: 2016-12-26

自查漏洞和漏洞修复提供便利. 实验证明, 该工具可以高效的检测和利用 Docker Remote API 未授权访问漏洞.

1 Docker 简介

Docker[3]是一个能够把开发的应用程序自动部署到容器的开源引擎, 目前项目代码在 GitHub 上进行维护[4]. Docker 在虚拟化的容器执行环境中增加了一个应用部署引擎. 该引擎的目标就是提供一个轻量、快速的环境, 能够运行开发者的程序, 并方便高效地将程序从开发者的笔记本部署到测试环境, 然后再部署到生产环境.

1.1 Docker 架构

Docker 使用客户端-服务器(client-server)架构模式. Docker 客户端会与 Docker 守护进程进行通信. Docker 守护进程会处理复杂繁重的任务例如建立、运行、发布 Docker 容器. Docker 客户端和守护进程可以运行在同一个系统上, 也可以使用 Docker 客户端去连接一个远程的 Docker 守护进程. Docker 客户端和守护进程之间通过 socket 或者 RESTful API 进行通信[5]. Docker Client 作为一个通信客户端用来与 Docker Daemon 守护进程进行通信, 从而现实对 Docker Container 进行管理[6], 如图 1 所示.

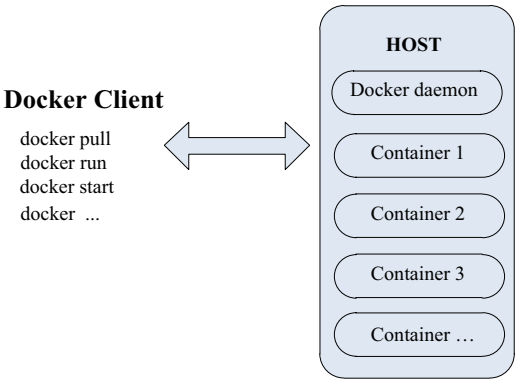

图 1 Docker 架构模式

1.2 Docker Remote API 与 DockerSwarm

Docker Remote API 是取代 rcli(远程命令行界面)的 REST API. 它有助于发出请求、获取与发送数据并且检索信息.

Swarm 是 Docker 公司在 2014 年 12 月初发布的一套较为简单的工具, 用来管理 Docker 集群, Swarm 将一群 Docker 宿主机变成一个单一的、虚拟的主机. Swarm 使用标准的 Docker API 接口作为其前端访问入口, 换言之, 各种形式的 Docker Client(Docker client in go, Docker_py, Docker 等)均可以直接与 Swarm 通信. Swarm 的结构图如图 2 所示.

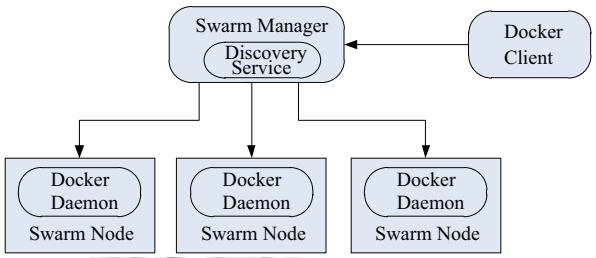

图 2 Swarm 的结构图

2 工具设计与实现

通过上文对 Docker Remote API 的介绍, 发现可以利用 REST API 接口对本地和远程的 Docker 程序及镜像和容器进行管理, 而且根据官方文档说明, 默认情况下该 API 接口是没有认证机制的. 同时进行集群配置时, 需要被管理的 Docker 节点上开放一个 TCP 端口(默认 2375)来与 Swarm manager 通信. 因此, 如果在公网机器上安装 Docker, 并且默认 2375 端口开放对外是非常危险的, 可以通过该 API 接口对远程主机进行命令执行操作, 进而可以直接写公钥进行远程登录管理操作. 本节通过介绍工具的设计模块说明工具是如何实现对 Docker Remote API 未授权访问漏洞的检测和利用.

2.1 漏洞检测模块

在本地搭建了两台虚拟主机, 一台为 Docker 的客户端, 另一台为 Docker daemon 的服务器, 开放 Remote API 端口 2375, 具体 IP 地址为: Docker-client: 192.168.79.134; Docker-server: 192.168.79.133. 首先, 进行漏洞检测, 具体流程如图 3 所示.

(1) 检测 Docker 的 API 端口是否开放

发现目标开放 2375 端口, 如图 4 所示.

(2) 利用 Docker 客户端连接检测是否存在未授权访问漏洞

可以通过该 API 接口正常查看 Docker 相关信息, 如图 5 所示, 故漏洞存在.

2.2 容器利用模块

检测到漏洞存在, 即可利用 Docker Remote API 执行 Docker 命令. 可以访问 http://host:2375/containers/

json 返回服务器当前运行的 container 列表, 可以创建/删除 container, 拉取 image 等. 如果当前运行的 container, 或者 image 内有代码或者其他敏感信息, 就可以继续渗透, 容器利用流程图如图 6 所示.

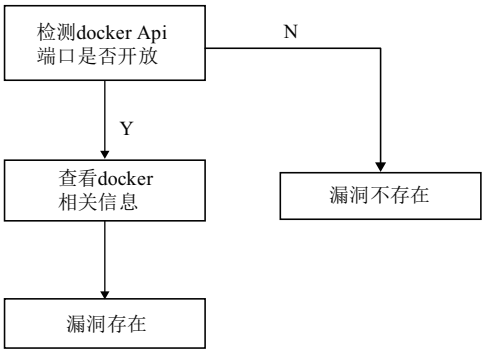

图 3 漏洞检测流程图

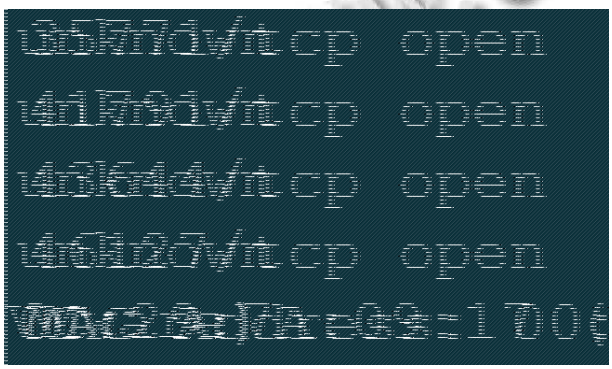

图 4 检测 Docker 的 API 端口

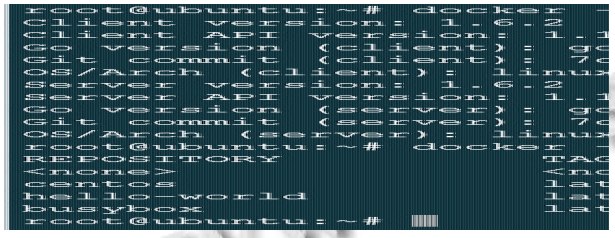

图 5 Docker 相关信息

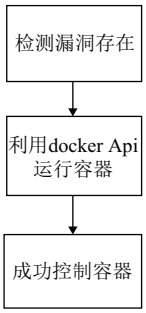

图 6 容器利用流程图

2.3 宿主机利用模块

由于通过 Docker 的 API 只能对 Docker 中的容器进行操作, 而无法控制宿主机器. 考虑到 Docker 是以 root 权限运行的, 虽然 Docker 执行的命令只能在容器内部, 与宿主机是隔离的, 但是通过上文 API 文档的查看, 在 Docker 运行容器的时候, 是可以将本地文件或目录作为 Volume 挂载到容器内部, 并且在容器内部, 这些文件和目录是可以修改的. 下文通过挂载写公钥的方式来进行漏洞利用, 具体流程如图 7 所示.

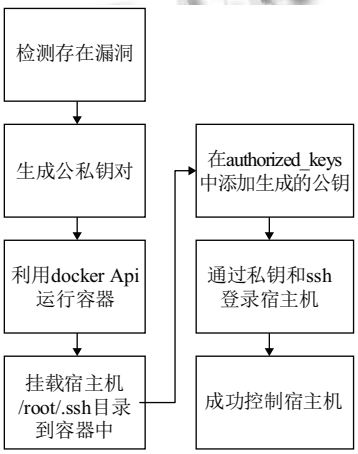

图 7 宿主机利用流程图

(1) 公私钥对的准备

公私钥的生成方式有很多, 可以在 xshell 中生成, 也可以直接在 linux 系统用 ssh-keygen 命令生成, 本文直接在系统中用命令生成, 如图 8 所示.

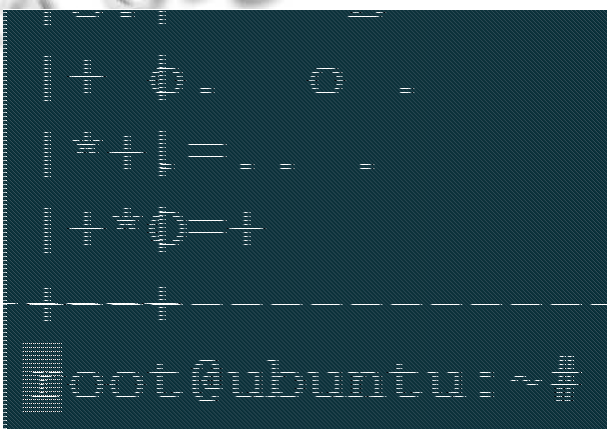

图 8 公私钥的生成

其中生成的公钥内容如图 9 所示.

(2) 利用 API 运行容器并挂载宿主机的/root/.ssh 目录到容器中

通过 Docker 客户端连接 API 接口并运行相应的镜像, 同时挂载宿主机的/root/.ssh 目录至容器的/mnt 目录下, 如图 10 所示.

图 9 生成的公钥

图 10 挂载宿主机的/root/.ssh 目录到容器中

进入到容器之后查看/mnt 目录, 发现宿主机的.ssh 目录已经挂载, 然后再在 authorized_keys 添加上刚刚新创建的公钥内容, 如图 11 所示.

图 11 新创建的公钥内容

(3) 通过 ssh 和私钥登录 Docker 宿主机

登录成功, 如图 12 所示, 即可对宿主机器进行任意操作.

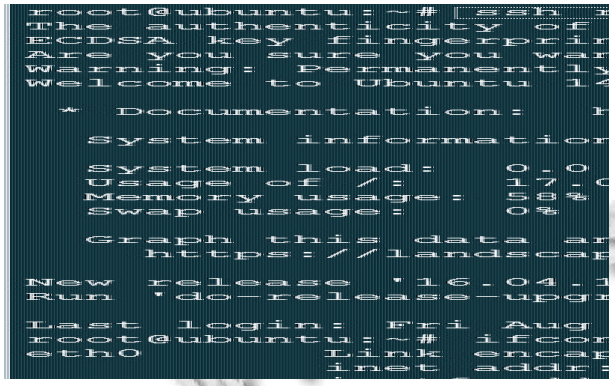

图 12 通过 ssh 进行无密登录 Docker 宿主机

3 工具测试及结果分析

3.1 工具测试

利用上文搭建的环境进行工具的测试. 输入测试地址 http://192.168.79.133:2375, 点击验证, 发现目标地址存在漏洞, 如图 13 所示.

点击利用即可进入漏洞利用界面, 此时有两个标签, “宿主机利用”和“容器利用”, 初次进入该页面, 下面的“结果显示”里的内容是当前可用的镜像信息.

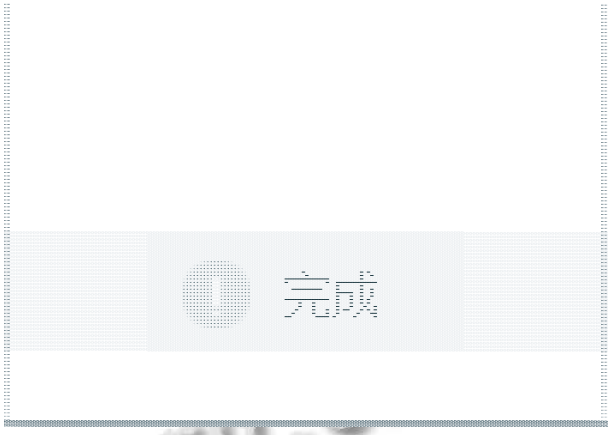

图 13 Docker Remote API 漏洞检测

选择可用的镜像 ID, 然后将准备的公钥信息复制到“公钥”框中, 再指定要使用私钥的位置, 点击“写入公钥”按钮后, 就将该公钥写入宿主机的 authorized_keys 文件中了. 后续在命令框中执行命令的过程即为利用私钥进行免密登录宿主机, 将命令执行结果返回“结果显示”中, 如图 14 所示.

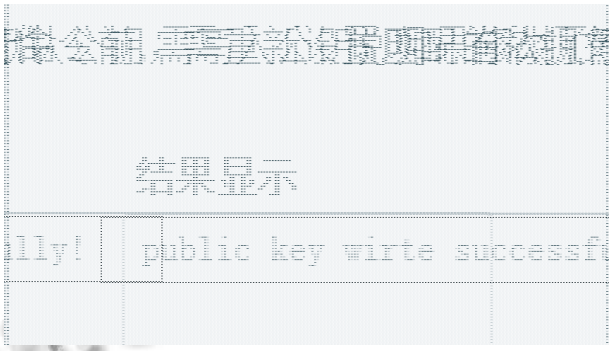

图 14 将公钥写入宿主机

公钥写入成功, 然后执行命令 whoami, 如图 15 所示.

图 15 宿主机利用命令执行

进入“容器利用”标签, 进行命令执令, 结果如图 16 所示.

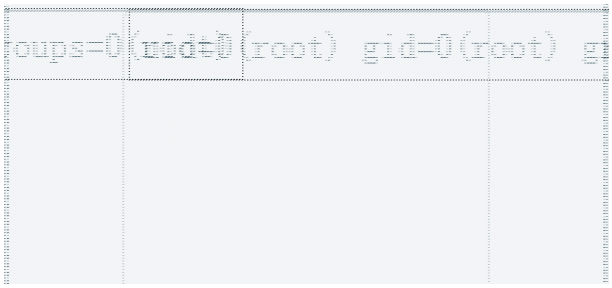

图 16 容器利用命令执行

3.2 结果分析

目前网络上主流的漏洞扫描工具大部分不提供 DockerApi 漏洞检测功能, 部分漏扫设备的最新版本已包含该漏洞的检测, 但是大部分漏扫设备部署代价昂贵, 不具备普及性, 具体对比如表 1 所示. 因此, DockerApi 漏洞利用工具方案弥补了目前该漏洞利用工具的缺失, 为漏洞的检测和修复提供了便捷, 同时该工具可以准确的检测出此类漏洞, 并具有耗时短的优势.

表 1 与网络上主流的漏洞扫描工具对比

漏洞扫描工具	能否检测出 DockerApi漏洞	检测出漏洞所占比例	平均耗时(s)
Acunetix Web Vulnerability Scanner	否	0/20	58
WebVulnScan	否	0/20	119
Nessus	否	0/20	242
启明星辰天境漏扫	是	18/20	305
绿盟Web应用漏洞扫描	是	19/20	327
DockerApi漏洞利用工具	是	20/20	3

4 结束语

本文通过对 Docker Remote API 未授权访问漏洞的产生、检测及利用技术的研究, 设计并实现了针对 Docker Remote API 未授权访问漏洞的检测利用工具. 该工具弥补了当前此漏洞检测工具的缺失, 并提供批量检测功能, 为漏洞的防护提供了便捷.

检测仅仅是漏洞修复的第一步, Docker Remote API 未授权访问漏洞的主要原因是由于 Docker API 在接收到命令参数时并没进行授权认证的过程, 而是直接对其进行了处理. 要想彻底解决这一类型的安全问题, 一是要增加访问控制措施, 对 API 端口做网络访问控制或者白名单; 二是要添加认证方式, 使用 TLS 认证, 令 Docker CLI 在发送命令到 Docker daemon 之前, 首先发送它的证书, 如果证书是由 daemon 所信任的 CA 签发, 才可以继续执行.

参考文献

1 2017 internet security threat report. https://www.symantec.com/security-center/threat-report.

2 Yeo J. Using penetration testing to enhance your company’s security. Computer Fraud & Security, 2013, 2013(4): 17–20.

3 Docker, Inc. What is Docker. http://www.docker.com/whatisdocker. [2015-01-29].

4 Seo KT, Hwang HS, Moon IY, *et al*. Performance comparison analysis of Linux container and virtual machine for building cloud. Advanced Science and Technology Letters, 2014, (66): 105–111.

5 Docker, Inc. Understand what are the major Docker components. https://docs.docker.com/introduction/understanding-docker. [2015-03-21].

6 孙宏亮. Docker 源码分析(一): Docker 架构. http://www.infoq.com/cn/articles/docker-source-code-analysis-part1. [2014-09-25].