哈尔滨工业大学深圳校区

毕业论文（设计）中期报告

题　　目　面向云原生的内核威胁检测系统的设计与实现

姓　　名　丁浩卓

学　　号　190110629

学　　院　计算机科学与技术学院

专　　业　计算机科学与技术

指 导 教 师　刘川意

日　　期

目　录

1 课题主要研究内容及进度

1.1 课题主要研究内容

本课题将从以下几个方面开展研究：

针对云原生环境下面临的风险进行威胁建模。针对常见安全漏洞，黑客攻击行为，恶意病毒样本总结分析攻击技术手段，提取技术特征。主要有以下两个方面：一是监控容器生命周期各阶段行为。检测系统支持在 Docker 容器启动、运行的各个阶段的对容器行为进行监控，可识别容器异常启动和容器运行时异常。二是研究 Rootkit 病毒行为特征。在内核函数调用层面对病毒进行运行时动态分析。

入侵检测系统的总体设计与实现。通过内核态eBPF程序实现内核运行时的行为监控，并将相关信息通过BPF系统调用接口传递到用户态的检测引擎，一旦命中规则即触发系统预警。

对入侵检测系统进行功能测试和性能评估，提高容器监控与 Rootkit 检测效率。将本系统与主流的 HIDS 系统进行性能对比，主要从系统功能、系统性能两个纬度进行比较。系统功能测试主要包括进程监控检测功能测试以及入侵行为检测测试。系统性能测试主要包括时延、带宽测试两大部分。针对性能瓶颈提高系统性能。

1.2 进度介绍

目前已经完成：

1. 2022 年 11 月，完成技术背景调研。
2. 2022 年 12 月，完成容器逃逸及 Rootkit 攻击的威胁分析，并成功提取恶意攻击的行为特征。
3. 2023 年 1 月至 3 月，入侵检测系统总体设计与实现，基本实现入侵检测功能。

正在收集并分析真实病毒样本，完善入侵检测系统功能。

2 已完成的研究工作及结果

2.1 威胁分析

针对云原生场景的安全威胁主要有两个方面：一是容器逃逸突破安全隔离，二是突破后植入 Rootkit 进行权限维持。

2.1.1 容器逃逸

容器逃逸指攻击者通过劫持容器化业务逻辑，或直接控制（CaaS 等合法获得容器控制权的场景）等方式，已经获得了容器内某种权限下的命令执行能力；攻击者利用这种命令执行能力，借助一些手段进一步获得该容器所在直接宿主机（经常见到“物理机运行虚拟机，虚拟机再运行容器”的场景，该场景下的直接宿主机指容器外层的虚拟机）上某种权限下的命令执行能力。

容器生命周期中三个过程的安全：构建时安全、部署时安全、运行时安全。在容器生命周期的三个过程中，攻击者往往是在前两个阶段部署相关的恶意代码，在容器运行时对环境真正执行相关的攻击指令。因此，容器运行时相比于其他两个阶段更直接、也更容易分析出环境中的恶意行为。与其他虚拟化技术类似，逃逸也是针对容器运行时存在的漏洞最为严重的攻击利用行为。攻击者可通过利用漏洞“逃逸”出自身拥有的权限范围，实现对宿主机或者宿主机上其他容器的访问，其中最为简单的就是造成宿主机的资源耗尽，往往会直接危害底层宿主机和整个云原生系统的安全。根据风险所在层次的不同，可以进一步展开为：危险配置导致的容器安全风险、危险挂载导致的容器安全风险、相关程序漏洞导致的容器安全风险、内核漏洞导致的容器安全风险：

1. 危险配置导致的容器安全风险：用户可以通过修改容器环境配置或在启动容器时指定参数来改变容器的相关约束，但如果用户为一些不完全受控的容器配置了某些危险的配置参数，就为攻击者提供了一定程度的可以攻击利用的安全漏洞，例如未授权访问带来的容器安全风险，特权模式运行带来的容器安全风险。
2. 危险挂载导致的容器安全风险：将宿主机上的敏感文件或目录挂载到容器内部，尤其是那些不完全受控的容器内部，往往也会带来安全风险。这种挂载行为可以通过环境配置来设定，也可以在运行时进行动态挂载，因此这里单独地归为一类。随着应用的逐渐深化，挂载操作变得愈加广泛，甚

至为了实现特定功能或方便操作，使用者会选择将外部敏感资源或文件系统直接挂载入容器，由此而来的安全问题也呈现上升趋势。例如：挂载 Docker Socket 引入的容器安全风险、挂载宿主机 procfs、sysfs 引入的容器安全问题等。

3. 相关程序漏洞导致的容器安全风险：所谓相关程序漏洞，指的是那些参与到容器运行、管理的服务端以及客户端程序自身存在的漏洞。例如，CVE-2019-5736、CVE-2021-30465、CVE-2020-15257 等存在于 Container Daemon、runC 上的容器安全漏洞。

4. 内核漏洞导致的容器安全风险：Linux 内核漏洞的危害之大、影响范围之广，使得它在各种攻防话题下都占有一席之地，特别是在容器环境中由于容器与宿主机共享了内核，攻击者可以直接在容器中对内核漏洞进行利用攻击。近年来，Linux 系统曝出过无数内核漏洞，例如最有名气的漏洞之一——脏牛（CVE-2016-5195）漏洞也能用来进行容器逃逸。

2.1.2 Rootkit 植入攻击

Rootkit 是一种特殊的恶意软件，它的功能是在安装目标上隐藏自身及指定的文件、进程和网络链接等信息，比较多见到的是 Rootkit 一般都和木马、后门等其他恶意程序结合使用。具有隐蔽性、持久性的特点。一旦 Rootkit 成功植入系统，用户将无法感知恶意后门的存在，传统手段难以将其清除。

Rootkit 可以根据其运行时的系统环境进行分类：

1. 用户态：通常基于 LD_PRELOAD 的预加载机制。

2. 内核态：通常基于 LKM 加载恶意内核模块。

3. 基于更底层的硬件、虚拟化管理软件实现：基于 Firmware 固件（UEFI）、基于 Hypervisor 中间软件层等。

本项目主要针对前两种 Rootkit 攻击，内核态及用户态 Rootkit 的植入。User-mode Rootkit（用户态 Rootkit）基于 Linux 动态库预加载机制，使用环境变量 LD_PRELOAD 和/etc/ld.so.preload、/etc/ld.so.conf 等加载黑客自定义的恶意库文件来实现后门功能。Kernel-mode Rootkit（内核态 Rootkit）通过可加载内核模块（LoadableKernel Module，LKM）将恶意代码直接加载进内核中。在攻击过程中，其中最关键的技术为 hook 函数，也常被称为钩子函数。这是一种用于改变系统行为的技术，内核态 Rootkit 主要就通过 hook 技术劫持内核中的函数，改变其原有

行为，最终实现权限维持及后门功能。

对于在内核中劫持函数的方式基于特性可以主要分为两大类：

1. 基于内核提供的跟踪框架及内核热补丁机制实现函数劫持，如基于 ftrace 跟踪框架的 Ftrace Hook 可轻松劫持系统调用表及内核函数。

2. 基于内核指令操作函数自实现的函数劫持。将目标函数对应内存中的前几个字节，替换成一个跳转指令。让函数开始执行的时候跳转到自定义的代码部分并执行，最后回到被劫持函数的正常执行逻辑。

2.2 入侵检测系统总体设计与实现

本文的入侵检测系统总体架构设计如图 2-1 所示，系统架构主要分为两个模块：内核实时监控模块，威胁规则匹配模块。

内核实时监控模块通过 eBPF 实现对内核运行时的细粒度监控。通过载入 eBPF 程序，在关键内核函数进行插桩，从而监控运行时的进程上下文，对系统调用及内核函数进行参数检查。将内核运行时信息基于 eBPF Map 以共享内存的方式传递出去。

威胁规则匹配模块主要由特征规则库及匹配引擎两部分组成。特征规则库主要基于对主流攻击手段的分析。匹配引擎将 eBPF 程序共享的系统运行时上下文信息与规则库进行规则匹配，匹配成功则发出警告。

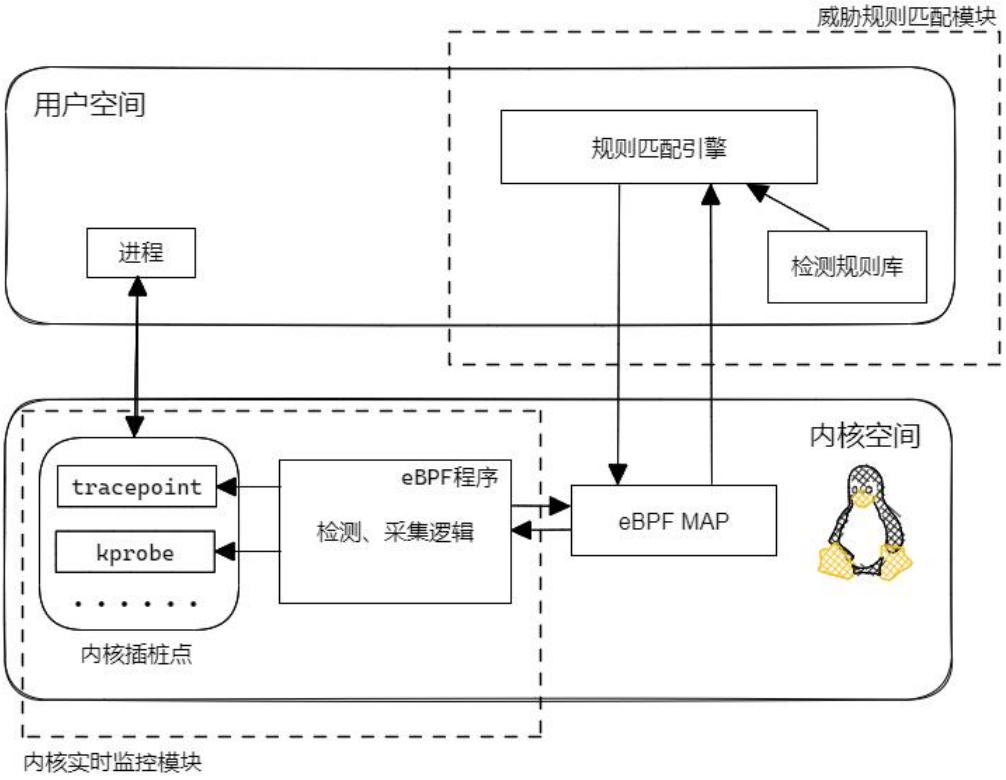

图 2-1　系统总体架构图

2.2.1 eBPF 原理

如图 2-2 所示，编写的 eBPF 程序首先被编译成 eBPF 字节码，通过 BPF 系统调用将编译好的 eBPF 程序载入内核。在载入内核的过程中，先由验证器（eBPF Verifier）进行安全检验，然后根据当前系统的硬件信息判断是否进行即时编译（JIT），即时编译会将 eBPF 字节码编译成对应的机器码，然后缓存起来。无法进行即时编译的 eBPF 程序会运行在内核中的 BPF 虚拟机中，虚拟机需要将 eBPF 字节码解析成机器码。即时编译之后，运行时直接执行机器码，提高效率。

eBPF 程序附着到 kprobe、tracepoints 等各类插桩点上，基于事件触发运行，对各类事件进行监听。运行过程中，用户空间与内核空间可以通过 eBPF Map 进行双向数据交互，本质上是一种共享内存的技术。

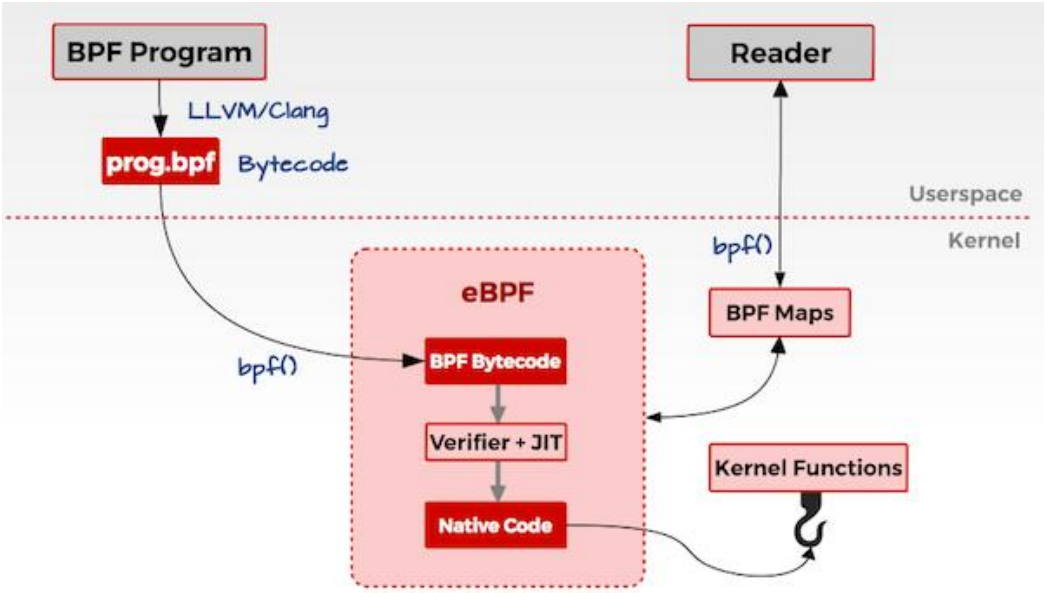

图 2-2　eBPF 原理图

2.2.2 内核实时监控模块

内核实时监控模块由 eBPF 程序实现，本系统主要基于 libbpf 开发 eBPF 程序。主要插桩（hook）的函数如表 2-1 所示，主要有系统调用、内核函数两大类。通过对关键内核函数调用的监控，实现了内核运行时的细粒度信息收集，提供了全面丰富的运行时信息。

表 2-1　内核插桩点列表

插桩点类型	插桩点名称
tracepoint	module_load
tracepoint	sys_exit_finit_module
tracepoint	sys_enter_mount

tracepoint	sys_exit_mount
tracepoint	sys_enter_open
tracepoint	sys_exit_open
tracepoint	sys_enter_openat
tracepoint	sys_exit_openat
tracepoint	sys_enter_execve
tracepoint	sys_enter_execveat
tracepoint	sys_enter_kill
kretprobe	kprobe_lookup_name
kretprobe	arm_kprobe
kretprobe	insn_init
kretprobe	insn_get_length

2.3 功能测试

2.3.1 容器逃逸预警

在基于 Cgroup 机制实现进程运行环境区分的基础之上，进一步实现容器异常行为检测，最终实现容器逃逸预警功能。目前的容器逃逸预警功能主要针对容器运行时安全的两个方面：危险配置导致的容器安全风险、危险挂载导致的容器安全风险。检测结果如下：

1. 危险配置导致的容器安全风险，如以高权限启动容器，检测结果如图 2-4。

```
dhz@ubuntu:~$ sudo docker run -it --privileged ubuntu:20.04
[sudo] password for dhz:
root@0d85acbc0ef6:/#
```

图 2-3 高权限启动容器容器命令

```
dhz@ubuntu:~/workspace/HIDS-eBPF/hids$ sudo ./hids
=============================================
TIME     EVENT               COMM          PID     PPID    PID_NS      DESCRIBE
04:18:52 MODULE_LOAD         modprobe      6066    5952    4026531836  load module, module-name is veth !
04:18:52 INSERT_MODULE_FINISH modprobe     6066    5952    4026531836  insert module finished, module-name is veth !
04:18:57 [Container start]    runc:[2:INIT] 6132    6104    4026532633  container-id: 0d85acbc0ef6, cap_effective:3fffffffff , The privileged container start
container is: 0d85acbc0ef693d94cafcb965b32bc3c14e423e2276935580d4d261c951ef23d
```

图 2-4 容器启动权限检查结果

2. 危险挂载导致的容器安全风险，如运行容器时将敏感目录挂载到容器内，检测结果如图 2-6。成功检查到容器的高危敏感目录挂载行为。

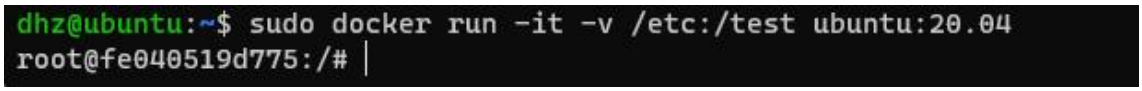

图 2-5 容器启动时挂载敏感目录命令

图 2-6 高危敏感目录挂载行为检测结果

2.3.2 Rootkit 植入检测

目前实现了对使用两类不同劫持内核函数技术的内核态 Rootkit 的检测及用户态 Rootkit 的检测，检测结果如下：

1. 直接修改系统调用表，实现对系统调用的劫持。Rootkit 实例：Diamorphine 。检测结果如图 2-7，成功检测出一系列恶意行为：恶意模块 diamorphine 加载、篡改内存、修改的系统调用表。

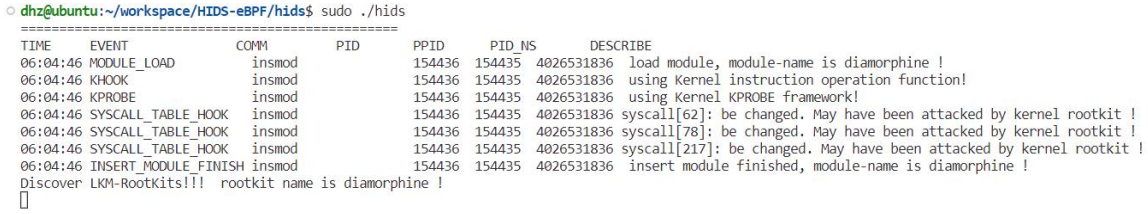

图 2-7 Diamorphine 检测结果

2. 基于 ftrace framework 实现对系统调用及内核函数的劫持。Rootkit 实例：brokepkg。检测结果如图 2-8，成功检测出一系列恶意行为：恶意模块 brokepkg 加载、使用内核 kprobe 框架、篡改内存。

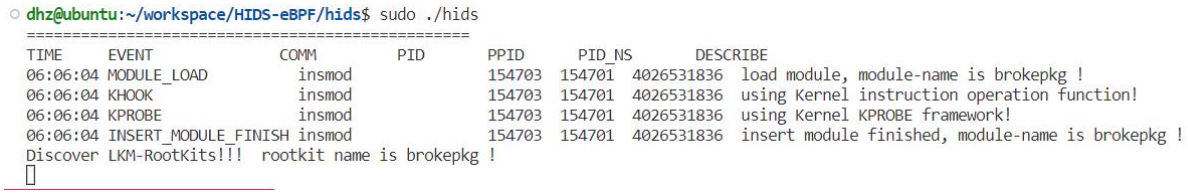

图 2-8 brokepkg 检测结果

3. 基于 Linux 动态库预加载机制的用户态 Rootkit，攻击实例：Azazel。检测结果如图 2-9，成功检测出当前系统环境异常：打印出被劫持的动态库函数。

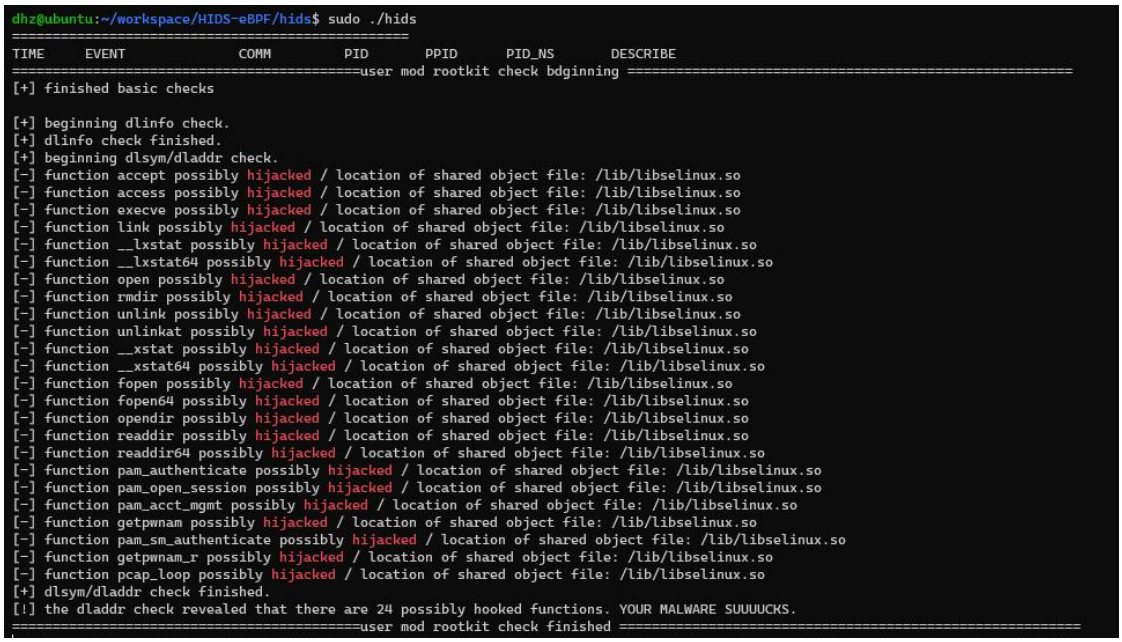

图 2-9　Azazel 检测结果

3 后期拟完成的研究工作及进度安排

3.1 后期拟完成的研究工作

1. 完善系统功能，继续收集分析 Rootkit 病毒实例，完善威胁检测规则库。
2. 在当前容器逃逸研究基础之上，继续深入进行容器逃逸分析。
3. 对入侵检测系统进行性能测试。

3.2 进度安排

具体进度安排如下：

2022 年 4 月 1 日——2022 年 4 月 15 日：收集分析 Rootkit 病毒实例，完善威胁检测规则库。

2022 年 4 月 16 日——2022 年 4 月 26 日：深入进行容器逃逸分析，完善容器逃逸检测功能。

2022 年 4 月 27 日——2022 年 5 月 15 日：对入侵检测系统进行性能测试。

2022 年 5 月 16 日——2022 年 6 月 20 日：撰写论文。

4　存在的困难及解决方案

4.1　存在的困难

1. 当前 Rootkit 攻击实例较少，且大部分 Rootkit 攻击需要满足特定的系统环境要求，如特定的内核、软件版本，搭设实验环境需要一定的时间。

2. 容器逃逸分析研究遇到瓶颈，进展缓慢。

4.2　解决方案

针对目前存在的问题和困难，提出以下解决方案：

1. 在 github 上继续收集分析 Rootkit 攻击实例，完善威胁检测规则库。

2. 继续学习研究容器逃逸相关案例及容器组件原理。

5 论文按时完成的可能性

本课题目前已经完成了：

1. 完成技术背景调研。完成容器逃逸及 Rootkit 攻击的威胁分析，并成功提取恶意攻击的行为特征。
2. 完成入侵检测系统总体设计与初步实现，基本实现入侵检测功能。

目前正在进行：

1. 继续收集分析 Rootkit 攻击实例，分析攻击特征。
2. 完善入侵检测系统功能。

综上所述，论文能够按期完成，并取得一定的研究成果。