浪潮信息云峦 KeyarchOS

安全防御组件 KSecure 测试指导

浪潮电子信息产业股份有限公司

2023 年 12 月

目 录

1 测试概述

1.1 应用场景

浪潮信息云峦服务器操作系统 KeyarchOS 提供轻量化的安全防御组件 KSecure，采用 eBPF 技术路线，提供主机安全检测和防御能力，在增强操作系统安全性和合规性的同时，解决传统内核模块方式带来的系统稳定性和性能问题。安全防御组件 KSecure 提供了合规性检测、关键文件/进程防护、主机入侵检测、勒索病毒诱捕等功能，主要应用场景如下：

- **安全合规**：基于标准和最佳实践对操作系统配置进行检测和自动修复，提升操作系统的合规性和安全性。支持用户结合实际的安全需求选择实施加固和扩展，以便更好地满足不同场景的配置安全基准要求
- **关键业务防护**：关键业务服务器提供安全防护，仅允许合法应用程序对关键业务文件进行访问；限制系统超级管理员权限，防止误操作或账号泄露导致的重要文件/配置的破坏。
- **防黑客入侵**：支持本地提权、Rootkit 攻击等入侵行为，及时发现并阻止勒索病毒加密行为，成为勒索病毒防护最后一道屏障。

1.2 技术背景

传统安全检测和防御方案采用内核模块技术，内核模块技术是通过编写内核模块来扩展操作系统的功能，内核模块可以直接访问和修改操作系统内核，可以实现高级别控制和丰富的功能，但编写不当的内核模块可能导致内核崩溃或引入安全漏洞。

eBPF 提供了一种安全、可编程的方式来扩展内核功能，eBPF 程序在内核中运行时会受到严格的安全限制，因此不会对系统的稳定性和安全性产生直接影响，可以实现深度的系统观测能力和自定义扩展能力。

1.3 测试内容

本文档是针对 KSecure 组件的安全特性进行测试，包括：合规性检测、访问控制、入侵检测和勒索病毒防护功能。

1.4 术语解释

名词	描述
eBPF	是英文 extended Berkeley Packet Filter 的缩写，它是一个能够在内核运行沙箱程序的技术，提供了一种在内核事件和用户程序事件发生时安全注入代码的机制，使得非内核开发人员也可以对内核进行控制。

2 软硬件环境

2.1 硬件

设备名称	部件	配置
NF5177M4	CPU	4*CPU
	内存	4G 内存
	硬盘	500G 硬盘

2.2 软件

软件	版本
KSecure	V1.0

2.3 测试工具

无

3 测试指导

3.1 合规性检测

KSecure 的合规性检测功能通过对操作系统的配置进行检测和修复，提高其配置的安全性和合规性，并实现了应急场景下的回退功能，为用户构造了一个更加安全的操作系统平台。

3.1.1 合规性检测

合规性检测支持按照模版检测，KSecure 组件基于等保要求和 CIS 标准内置合规性检测模版，检测类别包括：身份鉴别、入侵检测、访问控制、安全审计等。用户可以使用默认检测模版 basic，也可以根据需要自定义检测模版和基线值。

3.1.1.1 使用默认模版检测

管理员可通过 KSec baseline scan -t basic 命令（-t 参数指定使用的模版名称）来进行合规性检测，及时发现系统存在的配置安全风险。

```
[root@localhost opt]# KSec baseline scan -t basic
正在进行合规性检测......
```

图 3- 1-1 执行检测

检测执行完成后，命令行可直接查看本次检测结果，也可通过提示的检测结果路径进行查看。检测结果分为概览和检测详情两大部分，如下图所示：

```
[root@localhost opt]# KSec baseline scan
正在进行合规性检测......
检测完成，结果如下(也可查看结果文件/opt/KSec/compliance/log/scan_2023-11-29_22_16_43):
=================================================================================概览=================================================================================
检测时间：      2023-11-29 22:16:43
检测模板：      基础模板(/opt/KSec/compliance/template/basic)
基线配置：      /opt/KSec/policy/baseline.yaml
总检测项：      38
不通过项：      23
通过项：        15
通过率：        39.47%
=================================================================================检测详情=================================================================================
 序号     | 检测项                                                  | 基线值    | 实际值    |检测结果
 KOS1.1   | 检查密码复杂度策略-最小长度                             | 8         | 4         |fail
 KOS1.2   | 检查密码复杂度策略-大写字母个数                         | -1        | 未配置    |fail
 KOS1.3   | 检查密码复杂度策略-小写字母个数                         | -1        | 未配置    |fail
 KOS1.4   | 检查密码复杂度策略-数字个数                             | -1        | 未配置    |fail
 KOS1.5   | 检查密码复杂度策略-特殊字符个数                         | -1        | 未配置    |fail
 KOS1.6   | 检查是否设置密码散列算法为sha512                        | sha512    | sha512    |success
 KOS1.7   | 检查登录失败锁定处理策略-失败次数阈值                   | 5         | 5         |success
 KOS1.8   | 检查登录失败锁定处理策略-锁定时间(单位：秒)             | 600       | 300       |fail
```

图 3- 1-2 检测结果

基线检测详情字段具体如下：

No.	配置项	配置说明
1	序号	检测项的编号，对应检测模版文件和 yaml 配置文件中的序号
2	检测项	检测项的概要说明
3	基线值	检测项的检测基准值，对应 yaml 配置文件中的基线值
4	实际值	系统中该检测项的实际值，“未配置”表示系统配置文件中没有该项配置值
5	检测结果	基线值与实际值的对比结果，success 为检测通过，fail 为检测不通过

3.1.1.2 自定义检测模版

管理员也可通过新增/修改文件的方式自定义检测模版，模版内配置要检测的具体条目，通过修改检测模版文件（/opt/KSec/compliance/template）检测条目序号即可。

图 3- 1-3 模版文件

3.1.1.3 自定义基线值

当管理员基于业务实际场景想要修改检查基线值时，可以通过修改配置文件（/opt/KSec/policy/baseline.yaml）中对应项基线值的方式来自定义基线值，格式具体如下：

```
[root@localhost policy]# cat baseline.yaml
- type: 密码策略
  description: 检查系统密码策略是否合理
  items:
    - item:
        id: KOS1.1
        name: 检查密码复杂度策略-最小长度
        value: 8  基线值
        desp: 检测系统在创建用户密码时对密码长度的限制，密码长度越长，规则越复杂，被暴力破解的可能性越低。建议配置范围:大于等于8的数字
```

图 3- 1-4 基线值配置文件

3.1.2 合规性修复

管理员可通过 KSec baseline repair -t basic 命令（-t 参数指定使用的模版名称）来进行合规性修复，及时修复系统中存在的配置安全风险，减少黑客的攻击面。

```
[root@localhost policy]# KSec baseline repair -t basic
正在进行合规性修复......
```

图 3- 1-5 执行修复

修复执行完成后，命令行可直接查看本次修复结果，也可通过提示的结果路径进行查看，如下图所示：

```
正在进行合规性修复......
修复完成，结果如下(也可查看结果文件/opt/KSec/compliance/log/repair_2023-11-25_12_35_17):
=============================================================概览=============================================================
修复时间:       2023-11-25 12:35:17
修复模板:       基础模板(/opt/KSec/compliance/template/basic)
基线配置:       /opt/KSec/policy/baseline.yaml
总修复项:       24
失败项:         1
成功项:         23
成功率:         95.83%
=============================================================修复详情=============================================================
 序号     | 修复项                                  | 修复前          | 修复后          |修复结果
 KOS1.1   | 检查密码复杂度策略-最小长度             | 未配置          | 8               |success
 KOS1.2   | 检查密码复杂度策略-大写字母个数         | 未配置          | -1              |success
 KOS1.3   | 检查密码复杂度策略-小写字母个数         | 未配置          | -1              |success
 KOS1.4   | 检查密码复杂度策略-数字个数             | 未配置          | -1              |success
 KOS1.5   | 检查密码复杂度策略-特殊字符个数         | 未配置          | -1              |success
 KOS1.7   | 检查登录失败锁定处理策略-失败次数阈值   | 未配置          | 5               |success
 KOS1.9   | 检查密码更换周期策略-最大使用天数       | 99999           | 90              |success
 KOS1.11  | 检查密码更换周期策略-最小使用天数       | 0               | 7               |success
 KOS1.12  | 检查密码重用限制次数                    | 未配置          | 5               |success
 KOS1.13  | 检查密码策略是否对root生效              | false           | true            |success
 KOS2.3   | 检查SSH服务一次连接密码最大重试次数     | 6               | 4               |success
```

图 3- 1-6 修复结果

如有特殊业务需要，需修改检测和修复的基线值，可通过修改基线值 yaml 文件的方式进行修改，详情请参考 3.1.3 自定义基线值。

3.1.3 合规性回退

管理员可通过 KSec baseline rollback -t basic 命令（-t 参数指定使用的模版名称）来进行合规性回退，以便于在修复出现问题时能及时将系统还原回修复前的状态。

```
[root@localhost policy]# KSec baseline rollback -t basic
正在进行合规性回退......
```

图 3- 1-7 执行回退

回退执行完成后，命令行可直接查看本次回退结果，也可通过提示的结果路径进行查看，如下图所示：

```
正在进行合规性回退......
回退完成，结果如下(也可查看结果文件/opt/KSec/compliance/log/rollback_2023-11-25_12_36_04):
=============================================概览=============================================
回退时间：      2023-11-25 12:36:04
回退模板：      基础模板(/opt/KSec/compliance/template/basic)
基线配置：      /opt/KSec/policy/baseline.yaml
总回退项：      23
失败项：        0
成功项：        23
成功率：        100.00%
=============================================回退详情=============================================
序号    | 回退项                                          | 回退前    | 回退后    | 回退结果
KOS1.1  | 检查密码复杂度策略-最小长度                     | 8         | 未配置    | success
KOS1.2  | 检查密码复杂度策略-大写字母个数                 | -1        | 未配置    | success
KOS1.3  | 检查密码复杂度策略-小写字母个数                 | -1        | 未配置    | success
KOS1.4  | 检查密码复杂度策略-数字个数                     | -1        | 未配置    | success
KOS1.5  | 检查密码复杂度策略-特殊字符个数                 | -1        | 未配置    | success
KOS1.7  | 检查登录失败锁定处理策略-失败次数阈值           | 5         | 未配置    | success
KOS1.9  | 检查密码更换周期策略-最大使用天数               | 90        | 99999     | success
KOS1.11 | 检查密码更换周期策略-最小使用天数               | 7         | 0         | success
KOS1.12 | 检查密码重用限制次数                            | 5         | 未配置    | success
KOS1.13 | 检查密码策略是否对root生效                      | false     | false     | success
KOS2.3  | 检查SSH服务一次连接密码最大重试次数             | 4         | 6         | success
```

图 3- 1-8 回退结果

3.2 勒索病毒防护

勒索病毒是一种恶意软件，其主要目的是通过对用户数据加密，并向用户勒索赎金以获取解密密钥。勒索病毒通常通过电子邮件、垃圾邮件、社交媒体、文件共享等途径传播，一旦感染就会对系统和数据造成严重的安全威胁。

为了保护操作系统和数据的安全，KSecure 安全组件提供了勒索病毒防护功能，通过精准投放诱饵的方式实时监测潜在的勒索病毒威胁，及时发现和阻止恶意软件的入侵，避免对系统和用户数据进行破坏。

3.2.1 基本策略配置

KSecure 组件默认勒索病毒防护功能不启用，可以通过配置文件进行设置。默认配置文件如下图：

```
#策略基本信息
name: ransomware-protect-policy          #必填：策略名称
module: ransomware                       #必填：勒索病毒防护；禁止修改


#勒索病毒防护配置
switch-on: false                         #必填：勒索病毒防护开关,true-开启,false-关闭
kill-process: false                      #必填：可疑进程处置的动作,true-杀死进程,false-只拦截不杀死
#decoyFileDir:                           #可选：增加自定义诱饵文件投放目录(只生效前100个)
#  - dir: /root
#whiteList:                              #可选：白名单程序操作诱饵文件时,只阻断不杀死程序进程,且不记录日志(只生效前100个)
#  - path: /usr/bin/cat
```

图 3- 2-1 勒索病毒防护模块配置文件

勒索病毒防护模块配置文件详细字段如下表：

No.	配置项	名称	配置说明
1	switch-on	功能开关	勒索病毒防护功能开关。true 为启用，false 为关闭，默认为 false
2	kill-process	是否杀死可疑进程	对可疑进程处置动作。true 为直接杀死可疑进程，false 为监控不杀死
3	whiteList	白名单进程	对于业务程序对诱饵操作产生的误报，可以加入白名单。白名单中的程序操作诱饵文件不会被杀死，并且不记录日志。
4	decoyFileDir	自定义诱饵目录	可以根据实际需要在业务关键目录下投放诱饵

管理员可以根据需要进行配置，修改后通过 KSec policy add ransom.yaml 重新加载 yaml 文件后生效。

```
[root@localhost policy]# KSec policy add ransom.yaml
Success
```

图 3-2-2 加载勒索病毒防护策略

启用后安全组件会在系统关键位置投放诱饵文件，文件默认隐藏，文件内容中包含“此文件为勒索病毒诱饵文件，请勿操作！！！”字样，避免误删。

3.2.2 查看安全日志

管理员可通过查看日志文件来查看系统是否受到勒索病毒威胁或者检查是否存在误报。


```
{"time":"2023-11-29 22:32:27","action":"block","source":"/usr/bin/vim","path":"/home/april/.00a433e22e.docx","user":"root","pid":864373,"ppid":862898}
{"time":"2023-11-29 22:32:44","action":"block","source":"/usr/bin/bash","path":"/home/april/.ZZa433e22e.docx","user":"root","pid":862898,"ppid":862897}
{"time":"2023-11-29 22:32:44","action":"block","source":"/usr/bin/bash","path":"/home/april/.ZZa433e22e.docx","user":"root","pid":862898,"ppid":862897}
```


图 3- 2-3 勒索防护日志

勒索防护日志字段详细说明如下：

No.	字段名	说明
1	time	安全事件产生的时间
2	logType	事件类型，勒索病毒防护为 RansomWare
3	source	可疑勒索病毒来源，即进程全路径
4	user	操作的用户信息，例如（root）
5	pid	可疑进程 id
6	ppid	可疑进程的父进程 id
7	action	对可疑进程的处置动作，其中 block 为监控，kill 为终止进程

3.2.3 配置白名单程序

当产生误报时，管理员可通过在 yaml 文件中增加白名单程序的方式来消除误报，以避免误杀用户的业务进程。配置项为 whiteList，path 为进程全路径，支持配置多个，具体如下：

```
#策略基本信息
name: ransomware-protect-policy          #必填：策略名称
module: ransomware                       #必填：勒索病毒防护；禁止修改


#勒索病毒防护配置
switch-on: false                         #必填：勒索病毒防护开关,true-开启,false-关闭
kill-process: false                      #必填：可疑进程处置的动作,true-杀死进程,false-只拦截不杀死
#decoyFileDir:                           #可选：增加自定义诱饵文件投放目录(只生效前100个)
#  - dir: /root
#whiteList:                              #可选：白名单程序操作诱饵文件时,只阻断不杀死程序进程,且不记录日志(只生效前100个)
#  - path: /usr/bin/cat
```

图 3-2- 4 白名单进程配置

修改配置文件后，通过 KSec policy add ransom.yaml 命令重新加载 yaml 文件生效。

3.2.4 自定义诱饵目录

默认诱饵投放到操作系统重要目录下，为了能更灵活的保护用户的关键业务和关键数据，安全组件提供了自定义诱饵投放目录的功能。通过配置文件（ransom.yaml 的 decoyFileDir 字段）来增加诱饵投放目录，例如，业务的关键目录或者重要数据的目录。

```
#策略基本信息
name: ransomware-protect-policy          #必填：策略名称
module: ransomware                       #必填：勒索病毒防护；禁止修改


#勒索病毒防护配置
switch-on: true                         #必填：勒索病毒防护开关,true-开启,false-关闭
kill-process: false                      #必填：可疑进程处置的动作,true-杀死进程,false-只拦截不杀死
#decoyFileDir:                           #可选：增加自定义诱饵文件投放目录(只生效前100个)
#  - dir: /root
#whiteList:                              #可选：白名单程序操作诱饵文件时,只阻断不杀死程序进程,且不记录日志(只生效前100个)
#  - path: /usr/bin/cat
```

图 3-2- 5 自定义诱饵投放目录

修改配置后，通过 KSec policy add ransom.yaml 命令重新加载 yaml 文件生效，KSecure 组件会自动将诱饵投放到设置的目录下。

3.3 入侵检测

主机入侵检测功能模块监控系统中的文件操作、进程创建、网络连接等行为，通过入侵检测引擎进行判决，实现入侵事件识别。支持 Rootkit、本地提权、进程注入、反弹 shell、无文件攻击、内核模块加载、敏感文件读取等常见入侵行为检测，同时支持用户自定义检测规则。可通过 yaml 配置功能启用与关闭。

3.3.1 查看检测策略

KSecure 安全组件内置了 Rootkit、本地提权、进程注入等 10 类（18 条）常见入侵行为检测规则（/opt/KSec/policy/ ids.yaml），入侵检测规则文件内容包括两大部分：

1. **全局白名单**：用于过滤和处理误报，当正常业务或者日常维护操作被识别为入侵行为，可以将进程设置为全局白名单，详细配置方式见 5.4 设置白名单程序；

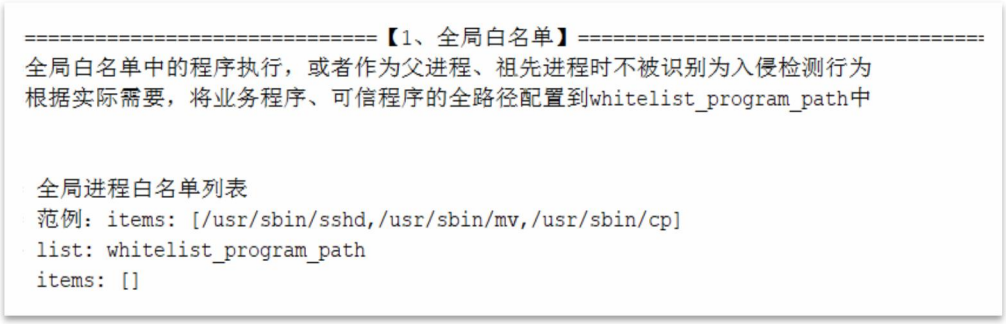

图 3-3- 1 全局白名单规则

2. **入侵检测规则**：内置了入侵检测规则，包括两部分：全局变量和入侵详细规则；

```
==============================【2.1 全局变量】==================================================
宏定义和列表被入侵检测规则使用，修改可能会导致规则失效。
list:列表，可以被规则(rule),macros(宏定义)和其他列表使用的集合
macro:宏定义，可重用的规则条件片段。在规则的判断条件和其他宏定义中使用

 macro: never_true
 condition: (evt.num=0)
 macro: open_write
 condition: (evt.type in (open,openat,openat2) and evt.is_open_write=true and fd.typechar='f' and fd.num>=0)
 macro: open_read
 condition: (evt.type in (open,openat,openat2) and evt.is_open_read=true and fd.typechar='f' and fd.num>=0)
 macro: rename
 condition: (evt.type in (rename, renameat, renameat2) and evt.res=SUCCESS)
 macro: create_hardlink
 condition: (evt.type in (link, linkat) and evt.dir=<)
 macro: create_symlink
 condition: (evt.type in (symlink, symlinkat) and evt.dir=<)
 macro: chmod
 condition: (evt.type in (chmod, fchmod, fchmodat) and evt.dir=<)
 macro: proc_name_exists
 condition: (proc.name!="<NA>")
 macro: spawned_process
 condition: (evt.type in (execve, execveat) and evt.dir=<)
 macro: container
 condition: (container.id != host)
```

图 3-3- 2 入侵规则全局变量

```
==============================【2,2 入侵检测规则】==================================================
规则结构如下
 rule:      规则名
 desc:      规则描述
 condition: 规则判断条件,这里定义了这条规则在什么条件可以被匹配到
 output:    规则匹配后，输出的日志内容
 priority:  规则优先级别从高到底为-emergency(紧急),alert(警报),critical(关键),error(错误),warning(警告)
 tags:      标签

 规则1: rootkit-检测/dev目录下文件创建
 macro: user_known_create_files_below_dev_activities
  condition: (never_true)
 rule: Create files below dev
  desc: >
    检测除授权的设备管理程序外/dev目录下文件的创建,这可以揭示rootkits隐藏在/dev目录下的文件。某些rootkit
  condition: >
    (evt.type = creat or (evt.type in (open,openat,openat2)))
    and evt.arg.flags contains O_CREAT
    and fd.directory = /dev
    and not built_in_whitelist
    and not global_whitelist_program
    and not proc.name in (dev_creation_binaries)
    and not fd.name in (allowed_dev_files)
    and not fd.name startswith /dev/tty
    and not user_known_create_files_below_dev_activities
  output: File created below /dev by untrusted program (file=%fd.name evt_type=%evt.type user=%user.name
  priority: ERROR
  tags: [maturity_incubating, host, filesystem, rootkit, mitre_persistence, T1543]
```

图 3-2- 3 入侵检测详细规则

入侵检测规则条目详情如下：

NO.	类别	规则名	检测规则说明

1	Rootkit 检测	Create files below dev	检测除授权的设备管理程序外 /dev 目录下文件的创建
2		Write below binary dir	尝试写入特定二进制目录下的任何文件，可以作为跟踪一般系统更改的审计规则。
3	内核模块加载	Linux Kernel Module Injection Detected	使用 insmod 或 modprobe 实现注入 Linux 内核模块
4	劫持执行流	Modify Ld preload file	检测 LD_PRELOAD 的使用.攻击者会使用这种技术来改变应用程序的行为或加载他们自己的程序
5	无文件执行	Fileless execution via memfd_create	使用 memfd_create 技术检测二进制文件是否从内存执行
6	进程代码注入	PTRACE attached to process	检测使用 PTRACE 向进程注入潜在恶意代码以逃避基于进程的防御或者提升权限的尝试
7		Proc Memory attached to process	检测到可能向另一个进程注入代码，攻击者可能会使用它来执行他们的恶意代码
8	计划任务检测	Schedule Cron Jobs	定时 cron 任务检测，利用 cron 的功能是入侵者使用的最古老的 TTPs(战术,技术,程序)之一
9	敏感文件读取	Read sensitive file untrusted	检测试图读取任何敏感文件(例如包括用户/密码/身份验证的文件)的行为
10		Create Hardlink Over Sensitive Files	检测在/etc 或根目录下敏感文件或子目录上创建硬链接
11		Create Symlink Over Sensitive Files	检测在/etc 或根目录下敏感文件或子目录上创建符号链接
12	本地提权	Non sudo setuid	检测通过使用 setuid 改变用户的尝试，sudo/su 除外
13		Set Setuid or Setgid bit	检测使用 chmod 设置 setuid 或 setgid 位
14		Sudoers file modification detected	检测 sudoers 文件被修改
15	反弹 shell	System procs network activity	检测反弹 shell，这是攻击者常用的技术手段之一
16	痕迹擦除	Clear Log Activities	检测关键访问日志文件的清除，通常是为了清除可归因于攻击者行为的证据
17	非法执行脚本/程序	Execution from /dev/shm	该规则检测/dev/shm 目录下的文件执行情况，这是入侵者经常使

			用的一种策略，用于存储可读、可写和偶尔可执行的文件
18		Launch Suspicious Network Tool on Host	检测在主机上启动网络工具(如 netcat、nmap、tcpdump、socat 等)

3.3.2 查看安全日志

管理员可通过入侵检测日志（/opt/KSec/log/intrusion_detection.log）查看系统是否产生过入侵事件或者产生过误报，日志详情如下：

图 3-2- 4 入侵检测安全日志

入侵检测安全日志常见字段详细说明如下：

No.	字段名	说明
1	process	进程名称
2	exe_flags	文件访问权限标志
3	command	进程命令行
4	proc_exepath	父进程名称
5	parent	可疑进程 id
6	evt_type	系统调用类别
7	user	系统登录用户名称
8	user_uid	系统登录用户的 uid

3.3.3 配置白名单程序

当正常业务或者日常维护操作被识别为入侵行为，即日志中出现误报时，可通过修改规则文件中全局进程白名单的方式对误报进行处理。例如：管理员发现 nginx 进程和 httpd 进程产生大量的入侵检测告警日志，为避免这两个进程产生的事件被识别为入侵事件，管理员可以将日志中的进程名称添加到进程白名单列表中，多个进程以逗号分隔，如下图：

图 3- 2-5 设置白名单进程

如有其他规则需要修改请联系技术支持工程师。

3.4 访问控制

访问控制功能包括进程防护和文件防护两大功能：

- **文件防护**：内置系统关键文件加固规则，防止系统关键文件被篡改和误删除，管理员也可以自定义文件防护规则；
- **进程防护**：进程防护功能包括进程保护和进程控制两个功能模块。进程保护可以保护核心业务进程不被恶意终止；进程控制又称为进程黑名单，黑名单的进程禁止运行，保护系统资源。

3.4.1 启用控制功能

KSecure 安全组件的访问控制功能默认不启用，如需启用可通过加载访问控制模块的方式进行启用。

3.4.2 策略配置

KSecure 安全组件访问控制策略（opt/KSec/policy/ac.yaml）共分为防御模式设置、进程防护和文件保护三部分。修改配置后需使用命令“KSec policy add ac.yaml”重新加载后生效。

3.4.2.1 策略基本设置

基本配置信息主要设置防御模式，其中：策略名称和模块名称为固定设置，不允许修改。默认配置如下图：

```
#============================【1、基本信息】=====================================================
name: access-control-policy                 #必选：策略名称；禁止修改
module: access_control                      #必选：访问控制；禁止修改
action: Monitor                             #必选：Monitor：监控模式，仅产生日志；Block：拦截模式，拦截并产生日志
```

图 3- 4-1 访问控制基本配置

访问控制功能的防御模式包括两种：

- **监控模式**：设置为 Monitor，监控模式下不对违规操作进行拦截，只记录违规日志。主要用于在前期部署后观察阶段和维护阶段；
- **拦截模式**：设置为 Block，拦截模式下拦截违规操作并记录日志，主要用于日常防护阶段。

3.4.2.2 进程防护策略

进程防护规则包括两大部分：

1. **进程控制**：又称为进程黑名单功能，黑名单中的进程不允许运行。当系统中发现恶意进程并处理后，为避免恶意进程再次启动，可以将恶意进程添加到进程黑名单规则中，添加后该恶意进程将无法启动。默认配置如下图：

```
# 2.1、进程控制（进程黑名单）规则
#(示例效果为：禁止test进程启动)
#processBlackList:
#   - path: /usr/local/test                  #必选，进程全路径
#     desc: 禁止test进程启动                 #可选，进程描述信息
```

图 3-4- 2 进程控制规则

2. **进程保护：**如需保护核心业务进程不被恶意终止，可将业务进程添加到进程保护规则中。默认配置如下图：

```
# 2.2、进程保护规则
#(示例效果为：禁止终止审计守护进程)
processProtectList:
  - path: /usr/sbin/auditd                    #必选，进程全路径（此为预定义规则审计进程，用户可根据需要进行调整）
    desc: 禁止终止审计守护进程                  #可选，进程描述信息
```

图 3-4- 3 进程保护规则

3.4.2.3 文件防护策略

文件防护包括两大部分：

- **预定义规则：**安全组件预置的对操作系统关键目录的保护，默认是关闭状态，如需开启，需将下图中 switch-on 开关设置为 true 后，通过命令 KSec policy add ac.yaml 重新加载策略后生效，详细配置如下图：

```
# 3.2、文件保护预定义规则
#文件保护预定义规则：目的是保护系统关键目录，用户可根据需要进行调整
preFileList:
  switch-on: false                            #文件保护预定义规则加载开关，true为开，false为关
  rules:
    - path: /usr/bin
      mode: rx
      desc: 禁止在系统目录中创建和修改可执行文件
    - path: /usr/sbin
      mode: rx
      desc: 禁止在系统目录中创建和修改可执行文件
```

图 3-4- 4 文件防护预定义规则

预定义规则配置项详细说明：

No.	字段名	说明
1	path	被保护的目录
2	mode	被保护目录开放的权限；r=读， w=写， x=执行， d=

		删除，c=创建，m=移动或重命名
3	desc	当前规则的描述信息

- **自定义规则：**允许用户根据自身的业务特点进行配置，可以将需要保护的业务目录添加到保护项 objPath 中，并将对保护项有权限的业务进程或进程目录添加到信任列表 fromSource 中，根据实际需要配置相应的 mode 权限值，样例如下：

```
# 3.1、文件保护自定义规则
#文件保护自定义规则：用户自定义信任项、保护项和权限
#(示例效果为：保护nginx文件和配置文件不被篡改，仅允许nginx自身进程和cat进行读和写)
fileProtectList:
  - objPath: /usr/share/nginx/html
    mode: rw
    fromSource:
      - subPath: /usr/bin/nginx
      - subPath: /usr/bin/cat
  - objPath: /etc/nginx/nginx.conf
    mode: rw
    fromSource:
      - subPath: /usr/bin/nginx
      - subPath: /usr/bin/cat
```

图 3-4- 5 文件防护自定义规则

上图中根据业务需要配置 nginx 业务保护

- objPath：被保护的目录或文件，样例中为 nginx 业务目录和配置文件，分别为/usr/share/nginx/html 和/etc/nginx/nginx.conf
- fromSource：信任的进程，仅信任的进程允许操作被保护的目录或文件，样例中为 nginx 自身进程和 cat 进程，分别为/usr/bin/nginx 和/usr/bin/cat
- mode：被保护项对信任进程开放的权限，r=读，w=写，x=执行，d=删除，c=创建，m=移动或重命名，all=所有权限，空或者无此字段=无权限

3.4.3 查看安全日志

管理员可通过查看日志文件（/opt/KSec/log/access_control.log）来查看系统中是否产生过访问控制的违规操作或者检查是否存在误报。

{"time":"2023-11-29 16:08:41","action":"Monitor","source":"/usr/bin/touch","path":"/usr/sbin/test","operation":"create","user":"root","pid":2389549,"ppid":2389506}
{"time":"2023-11-29 16:08:41","action":"Monitor","source":"/usr/bin/touch","path":"/usr/sbin/test","operation":"write","user":"root","pid":2389549,"ppid":2389506}
{"time":"2023-11-29 16:08:41","action":"Monitor","source":"/usr/bin/rm","path":"/usr/sbin/test","operation":"delete","user":"root","pid":2389506,"ppid":2389503}

图 3-4- 6 访问控制安全日志

访问控制安全日志字段具体如下：

No.	字段名	说明
1	time	安全事件产生的时间
2	action	当前事件产生时安全组件的防御模式：Block-拦截，Monitor-监控，对应规则文件基本信息中的 action
3	source	产生安全事件的可疑进程
4	path	可疑进程操作的文件路径
5	operation	可疑进程对文件做的操作
6	user	产生安全事件的用户
7	pid	安全事件可疑进程的进程 ID
8	ppid	安全事件可疑进程的父进程 ID

3.5 组件管理

3.5.1 服务管理

安全组件安装后会自动启动 KSec 服务，管理员可以通过命令行对 KSec 服务进行管理。

命令字	说明
help	查看帮助

start	启动 KSec 服务
stop	停止 KSec 服务
restart	重启 KSec 服务
status	查看 KSec 服务当前状态
rebooton	启用 KSec 服务的随系统自动启动功能
rebootoff	关闭 KSec 服务的随系统自动启动功能

3.5.2 资源限制

为了不影响系统和用户业务的正常运行，安全组件提供了限制自身资源占用的功能，可以通过配置文件（/opt/KSec/conf/KSec.yaml）中的 cpuLimit 来限制 KSecMain 进程占用的 CPU 资源（默认值为 25，可配置范围为 25 到 99）。修改后通过 KSec restart 重启生效。

```
[root@localhost conf]# cat KSec.yaml
gRPC: "32767"

#关键文件防护：on-功能开启,off-功能关闭。默认关闭
file_protect: off
#勒索病毒防护：on-功能开启,off-功能关闭。默认关闭
ransom: off
#入侵检测：on-功能开启,off-功能关闭。默认关闭
intrusion_detection: off

#cpu控制大小在25%-99%之间，不属于这个范围的按照默认值25%，单位默认为百分比
cpuLimit: 25    CPU限制
#日志等级，默认为INFO级别，可以设置INFO、DEBUG级别
```

图 3-5- 1 配置文件

4 测试用例

4.1 合规性检测

用例编号:	Function-4-1-1
测试目的:	以“密码复杂度”检测项为例，测试合规性检测功能
预置条件	密码复杂度检测项不符合要求，操作方法如下： 编辑/etc/pam.d/system-auth 文件，修改 password requisite pam_pwquality.so 行中的 redit=-1 lcredit=-1 dcredit=-1 ocredit=-1 minlen=6 中的前四个值的某一个值改为大于-1 或者 minlen 属性改为小于 8
测试步骤:	1. 进行合规性扫描，执行命令“Ksec baseline scan -t basic” 2. 查看命令行检测结果
预期结果:	检测结果为不合规，正确记录扫描报告
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

用例编号:	Function-4-1-2
测试目的:	以“密码复杂度”检测项为例，测试合规性修复功能
预置条件	密码复杂度检测项不符合要求，操作方法如下： 编辑/etc/pam.d/system-auth 文件，修改 password requisite pam_pwquality.so 行中的 redit=-1 lcredit=-1 dcredit=-1 ocredit=-1 minlen=6 中的前四个值的某一个值改为大于-1 或者 minlen 属性改为小于 8
测试步骤:	1. 进行合规性修复操作，执行“KSec baseline repair 命令” 2. 查看修复结果，查看 system-auth 文件修改为 redit=-1 lcredit=-1 dcredit=-1 ocredit=-1 minlen=8
预期结果:	修复成功，正确记录修复报告
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

用例编号:	Function-4-1-3
测试目的:	以“密码复杂度”检测项为例，测试合规性回滚功能
预置条件	密码复杂度检测项不符合要求，操作方法如下： 编辑/etc/pam.d/system-auth 文件，修改 password requisite pam_pwquality.so 行中的 redit=-1 lcredit=-1 dcredit=-1 ocredit=-1 minlen=6 中的前四个值的某一个值改为大于-1 或者 minlen 属性改为小于 8
测试步骤:	1. 进行合规性修复操作，执行“KSec baseline repair 命令”，查看修复结果 2. 执行合规性回滚操作“Ksec baseline rollback 命令”，查看回滚报告 3. 查看回滚后配置文件结果
预期结果:	回滚成功，各项值已回滚到修复前状态
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

用例编号:	Function-4-1-4
测试目的:	以“密码复杂度”检测项为例，测试自定义基线值功能
预置条件	密码复杂度检测项配置文件中密码复杂度为 8
测试步骤:	1. 自定义密码复杂度检测项的基线值，执行命令“vi /opt/KSec/conf/baseline.yaml”编辑 yaml 文件命令，密码长度配置项“minlen”设置为 12 2. 进行合规性扫描操作，执行“KSec baseline scan 命令”，查看扫描结果 3. 进行合规性修复操作，执行“KSec baseline repair 命令”，查看修复结果
预期结果:	扫描不合规，执行修复后成功修为 12
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

4.2 访问控制

用例编号:	Function-4-2-1
测试目的:	测试 Nginx 配置文件防篡改功能
预置条件	1. 防御模式为拦截模式 2. 已部署 nginx
测试步骤:	1. 设置 yaml 配置文件，编辑/opt/KSec/policy/ac.yaml 文件，配置内容如下： fileProtectList: - objPath: /etc/nginx/ #配置文件路径 fromSource: /usr/sbin/nginx #bin 目录 mode: r #权限只读” 2. 新加载配置文件，执行命令“KSec policy add /opt/KSec/policy/ac.yaml” 3. 手动修改 nginx 配置文件，nginx.conf，查看安全日志 access_control.log
预期结果:	修改配置文件失败，正确记录安全日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

用例编号:	Function-4-2-2
测试目的:	测试进程保护功能
预置条件	防御模式为拦截
测试步骤:	1. 设置yaml配置文件,执行命令“vi /opt/KSec/conf/ac.yaml”，将系统的审计进程设置为保护进程，防止被恶意终止，配置内容如下： “processProtectList: - path: /usr/sbin/auditd” 2. 重新加载配置文件，执行命令“KSec policy add /opt/KSec/conf/ac.yaml” 3. 查询 auditd 进程的 pid，尝试通过“kill-9 pid” 命令杀死 auditd 进程

	4. 查看/opt/KSec/log/目录下的安全日志 access_control.log
预期结果:	进程结束失败，正确记录安全日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

用例编号:	Function-4-2-3
测试目的:	测试进程控制功能（黑名单中的进程不允许执行），
预置条件	1. /usr/local/目录下存在一个可执行程序 test 2. 防御模式为拦截
测试步骤:	1. 设置 yaml 配置文件,执行命令“vi /opt/KSec/conf/ac.yaml”，将 test 程序加入黑名单，不允许启动，配置内容如下： “processBlackList: - path: /usr/local/test” 2. 重新加载配置文件，执行命令“KSec policy add /opt/KSec/conf/ac.yaml” 3. 尝试通过“./test” 命令启动进程 4. 查看/opt/KSec/log/目录下的安全日志 access_control.log
预期结果:	test 程序启动失败，正确记录安全日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备 注:	
测试人员:	

4.3 入侵检测

用例编号:	Function-4-3-1
测试目的:	测试进程注入行为检测
预置条件	1. 入侵检测功能开启，入侵检测功能已启用 2. 已准备好进程注入模拟程序 A
测试步骤:	1. 运行进程注入模拟程序 A 2. 查看/opt/KSec/log/目录下的安全日志 intrusion_detection.log

预期结果:	正确记录进程注入告警日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备　　注:	
测试人员:	

用例编号:	Function-4-3=2
测试目的:	测试内核模块加载行为检测
预置条件	入侵检测功能开启
测试步骤:	1. 使用 root 用户登录系统，在/usr/lib/modules 目录下查找 *.ko，找到可用 ko，例如 ab.ko 2. 加载内核模块 ab.ko，执行命令“insmod ab.ko” 3. 查看/opt/KSec/log/目录下的安全日志 intrusion_detection.log
预期结果:	正确记录内核模块加载告警日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备　　注:	
测试人员:	

用例编号:	Function-4-3-3
测试目的:	测试可疑网络工具执行检测
预置条件	1. 入侵检测功能开启 2. 准备可疑网络工具，例如：tcpdump、ncat、nc 等
测试步骤:	1. 使用可疑网络工具，执行“tcpdump”命令或“ncat”命令 2. 查看/opt/KSec/log/目录下的安全日志 intrusion_detection.log
预期结果:	正确记录可疑网络工具执行告警日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备　　注:	
测试人员:	

4.4 勒索病毒防护

用例编号:	Function-4-4-1
测试目的:	测试勒索病毒诱捕检测功能
预置条件	准备勒索病毒样本文件
测试步骤:	1. 启用防护开关，编辑配置文件“/opt/KSec/policies/ransom.yaml”，将 “switch-on”配置项设置为“on” 2. 加载安全策略，执行命令“KSec policy add /opt/KSec/policy/ransom.yaml” 3. 在系统上执行勒索病毒样本文件(样本病毒会加密系统部分文件) 4. 查看/opt/KSec/log/目录下的安全日志 ransom_detection.log
预期结果:	检测到勒索病毒，正确记录安全日志
测试结果:	☐ 通过 ☐ 部分通过 ☐ 不通过 ☐ 未测试
备　　注:	
测试人员:	

用例编号:	Function-4-4-2
测试目的:	测试检测勒索病毒后自动终止功能
预置条件	准备勒索病毒样本文件
测试步骤:	1. 启用防护开关，编辑配置文件“/opt/KSec/policies/ransom.yaml”，将 “switch-on”配置项设置为“on” 2. 设置检测到可疑进程后自动结束进程，上述配置文件中将“kill-process”配置项设置为“true” 3. 加载安全策略，执行命令“KSec policy add /opt/KSec/policies/ransom.yaml” 4. 在系统上执行勒索病毒样本文件(样本病毒会加密系统部分文件) 5. 查看/opt/KSec/log/目录下的安全日志

	ransom_detection.log
预期结果:	勒索病毒进程被终止，正确记录安全日志
测试结果:	□ 通过 □ 部分通过 □ 不通过 □ 未测试
备　　注:	
测试人员:	

5 分析与结论

综合上述来看，安全防御组件 KSecure 提供了合规性检测、关键文件/进程防护、主机入侵检测、勒索病毒诱捕等功能，通过 KSecure 组件可以提升操作系统的合规性和安全性。