

影响域分析工具

一、背景

当某个包更新时，可能会有其他包受这个包影响而功能异常，所以在系统更新后，我们需要知道更新的包可能会对哪些功能模块有影响，了解影响域信息。

二、需求

2.1 更新系统里面的某个包时，能自动识别出更新这个包对系统其他组件的影响。

2.2 影响的组件能自动识别为对应的系统功能模块。（比如：影响到了软件商店、文件管理等）

三、思考

3.1 影响域是什么

影响域的定义是包的反向依赖关系。

比如：C依赖B且B依赖A，那么A更新后可能对B的功能有影响，B功能异常又可能导致C功能异常，那么A的影响域就包括了B和C。

引申出来，影响域就是指包的反向依赖链。

3.2 如何知道新的iso相比上一个版本，哪些包变化了？

Build: (66456) Kylin-Desktop-V10-SP1-pangux-RC3-2503-Build01-20250307-arm64

基本状态 配置信息 **修改记录** 包列表

版本	体系结构	描述	完成时间
(66456) Kylin-Desktop-V10-SP1-pangux-RC3-2503-Build01-20250307 当前	arm64	【提测】 [pangux 2503-RC3] task 442447 【版本管理组】	2025-03-07 19:58:01
(66231) Kylin-Desktop-V10-SP1-pangux-RC2-2503-Build02-20250228	arm64	【提测】 [pangux 2503-RC2] task 436592 【版本管理组】	2025-02-28 17:36:50 与当前版本对比
(66195) Kylin-Desktop-V10-SP1-pangux-RC2-2503-Build01-20250227	arm64	【废弃】 [pangux 2503-RC2] task 436592 【版本管理组】	2025-02-27 19:50:50 与当前版本对比

kybuild平台上现在有接口，可以对比出两个iso的差异信息，结果为.xls 文档

文档中有<包列表对比>和<光盘源对比>等子表，子表中的'包名'列记录了所有更新包

3.3 如何获取包的影响域？

我们已知`apt-rdepends --state-follow=Installed --state-show=Installed -r "pkg"`该指令可以获取pkg这个包在该系统下的反向依赖关系（反向依赖关系也就是影响域）

但是该指令是在一个已安装的系统下执行，实际情况中我们没有条件在提测之前还要装好上一个系统，去跑这一条指令。

让我们把眼光看向kybuild平台，平台中有如下文件：

https://builder.kylin.com/kybuilder/build/getfile/{build_id}/casper/filesystem.squashfs

该squashfs文件约等于是这个iso装好后的环境，把这个文件挂载出来，可以获取到具体的文件系统。

那么我们是否可以通过这个挂载出来的文件系统，整理出该系统的依赖关系网呢？

答案是可以的！

/var/lib/dpkg/status 文件（以下简称status文件），该文件记录系统中的包信息。

```

1 Package: adduser
2 Status: install ok installed
3 Priority: important
4 Section: admin
5 Installed-Size: 798
6 Maintainer: Ubuntu Core Developers <ubuntu-devel-discuss@lists.ubuntu.com>
7 Architecture: all
8 Multi-Arch: foreign
9 Version: 3.118kylin2k11.0
10 Depends: passwd (>= 1:4.8.1-1kylin5k13), debconf (>= 0.5) | debconf-2.0,
11 polycoreutils (>= 2.2.1), libchuid (>= 1.0.10kylin1)
12 Suggests: liblocale-gettext-perl, perl, ecryptfs-utils (>= 67-1)
13 Conffiles:
14 /etc/deluser.conf 773fb95e98a27947de4a95abb3d3f2a2
15 Description: add and remove users and groups
16 This package includes the 'adduser' and 'deluser' commands for creating
17 and removing users.
18 .
19 - 'adduser' creates new users and groups and adds existing users to
20 existing groups;
21 - 'deluser' removes users and groups and removes users from a given
22 group.
23 .
24 Adding users with 'adduser' is much easier than adding them manually.
25 Adduser will choose appropriate UID and GID values, create a home
26 directory, copy skeletal user configuration, and automate setting
27 initial values for the user's password, real name and so on.
28 .
29 Deluser can back up and remove users' home directories
30 and mail spool or all the files they own on the system.
31 .
32 A custom script can be executed after each of the commands.
33 Original-Maintainer: Debian Adduser Developers <adduser@packages.debian.org>

```

以上是status文件中一个包的数据，包括了包名、状态、依赖等信息。

注意status文件中的Status: 字段，比如：

```

1 Status: deinstall ok config-files
2 Status: install ok config-files
3 Status: install ok installed

```

该字段表示该包在系统中的状态，dpkg -l 的第一列数据就是这个状态的缩写，ii 对应的 install ok installed。那么根据这个状态可以知道哪些包是安装到系统上的。

由此，我们可以通过以下方法整理出来依赖关系和被依赖关系：

①遍历整个status文件，识别关键字"Package: "、"Source: "、"Version: "、"Status: " 记录包名、源码包名字、版本号及其安装状态

②第二此遍历status文件，识别关键字"Pre-Depends: " 和 "Depends:"，截取依赖包，并检查依赖包的状态是否为"install ok installed"，如果是则计入依赖。在计入依赖的同时，将该Package 写入依赖包的反向依赖关系中。

```

1 def parse_status_file(status_file_path):
2     """解析status文件，提取包信息（分两步确保依赖解析完整）"""
3     packages = {}

```

```

4     try:
5         # 第一步：仅解析所有包的基本信息（名称、状态、版本、源码包）
6         with open(status_file_path, 'r') as file:
7             current_package = None
8             for line in file:
9                 line = line.strip()
10
11                # 遇到新包时初始化
12                if line.startswith('Package:'):
13                    parts = line.split(': ')
14                    if len(parts) < 2:
15                        logging.warning(f"格式错误的行: {line}")
16                        current_package = None
17                        continue
18                    package_name = parts[1].strip()
19                    current_package = {
20                        'name': package_name,
21                        'version': None,
22                        'source': package_name,
23                        'source_version': None,
24                        'depends': [],
25                        'state': None,
26                        'rdepends': []
27                    }
28                    packages[package_name] = current_package
29
30                # 提取版本信息
31                elif current_package and line.startswith('Version:'):
32                    parts = line.split(':', 1)
33                    if len(parts) >= 2:
34                        version = parts[1].strip()
35                        current_package['version'] = version
36                        current_package['source_version'] = version # 先默认
37
38                # 提取源码包信息
39                elif current_package and line.startswith('Source:'):
40                    parts = line.split(':', 1)
41                    if len(parts) >= 2:
42                        source_package = parts[1].strip()
43                        current_package['source'] = source_package
44                        # 提取源码包版本
45                        match = re.search(r'\((.*?)\)', line)
46                        if match:
47                            current_package['source_version'] =
48match.group(1)
49
50                # 提取状态信息
51                elif current_package and line.startswith('Status:'):
52                    parts = line.split(':', 1)
53                    if len(parts) >= 2:
54                        current_package['state'] = parts[1].strip()
55
56                # 第二步：单独解析所有包的依赖关系（此时所有包已在packages中）
57                with open(status_file_path, 'r') as file:
58                    current_package = None
59                    for line in file:
                        line = line.strip()

```

```

60
61         # 定位当前包
62         if line.startswith('Package:'):
63             parts = line.split(': ')
64             if len(parts) < 2 or parts[1].strip() not in packages:
65                 current_package = None
66                 continue
67             current_package = packages[parts[1].strip()]
68
69         # 解析依赖
70         elif current_package and line.startswith(('Depends:', 'Pre-
Depends:')):
71             parts = line.split(':', 1)
72             if len(parts) < 2:
73                 continue
74             depends = parts[1].strip()
75             dependencies = []
76             for dep_group in depends.split(', '):
77                 # 拆分可选依赖 (用|分隔)
78                 optional_deps = [dep.split(' ')[0].strip() for dep
in dep_group.split(' | ')]
79                 # 遍历可选依赖, 找到第一个已安装的依赖
80                 for dep in optional_deps:
81                     if dep in packages: # 此时所有包已解析, 直接判断是否
在packages中
82                         # 检查状态 (仅保留已安装的依赖)
83                         if packages[dep]['state'] == 'install ok
installed':
84                             dependencies.append(dep)
85                             # 记录反向依赖
86                             packages[dep]
['rdepends'].append(current_package['name'])
87                             current_package['depends'] = dependencies
88
89         except FileNotFoundError:
90             logging.error(f"未找到文件: {status_file_path}")
91             return {}
92         except Exception as e:
93             logging.error(f"解析status文件时出现错误: {e}")
94             return {}
95
96     return packages
97

```

由此, 可以整理出该系统的依赖关系图。

3.4 为什么通过status文件分析得到的关系比apt-rdepends 的结果少?

在系统执行xyh@xiaoyuhang:/data/work/单包更新影响域分析工具\$ apt-rdepends --state-follow=Installed --state-show=Installed -r kylin-system-updater

```

1 Reading package lists... Done
2 Building dependency tree
3 Reading state information... Done

```

```

4 kylin-system-updater
5   Reverse Depends: kcm (>= 3.9.0.0-0k0.9)
6   Reverse Depends: kylin-background-upgrade (>= 1.3.2.9-0k2.17update2)
7   Reverse Depends: kylin-installer (>= 1.1.86.1-0k0.7)
8   Reverse Depends: kylin-printer (1.3.1.0-0k1.10)
9   Reverse Depends: kylin-scanner (3.2.0.21-0k2.15)
10  Reverse Depends: kylin-software-center (5.0.6.8-0k1.28)
11  Reverse Depends: kylin-software-properties (>= 0.0.1.1-0k4.11)
12  Reverse Depends: kylin-update-frontend (>= 1.0.2.26-0k10.48)
13  Reverse Depends: preinstalled-apps (1.0.0.6-0k0.29)
14 kcm
15 kylin-background-upgrade
16   Reverse Depends: kylin-system-updater (>= 2.0.5.16-0k8.31)
17 kylin-installer
18   Reverse Depends: kylin-system-updater (>= 2.0.5.16-0k8.31)
19 kylin-printer
20 kylin-scanner
21 kylin-software-center
22   Reverse Depends: kcm (>= 4.0.0.0-0k0.4)
23 kylin-software-properties
24   Reverse Depends: kylin-background-upgrade (>= 1.3.2.9-0k2.17update2)
25   Reverse Depends: kylin-system-updater (>= 2.0.5.16-0k8.31)
26   Reverse Depends: ukui-control-center (>= 3.0.1-1-70.25update)
27 ukui-control-center
28   Reverse Depends: kcm (>= 3.9.0.0-0k0.9)
29   Reverse Depends: ky-miracast-source (>= 1.0.0.1-0k0.7)
30   Reverse Depends: kylin-nm (>= 3.24.0.0-0k2.24)
31   Reverse Depends: kylin-sso-client (2.2.11.1-0k0.23)
32   Reverse Depends: libkysdk-desktopctrl (>= 1.2.0.9-0k7.31)
33   Reverse Depends: libkysset (>= 1.1.0.1-0k1.9)
34   Reverse Depends: miraclecast (>= 1.0.8kord)
35   Reverse Depends: ukui-desktop-environment-core (2.0.3)
36   Reverse Depends: ukui-media (>= 3.24.0.0-0k1.8)
37 ky-miracast-source
38 kylin-nm
39   Reverse Depends: kcm (>= 3.9.0.0-0k0.9)
40   Reverse Depends: ksc-defender (>= 2.3.0.1-0k7.41)
41 ksc-defender
42 kylin-sso-client
43 libkysdk-desktopctrl
44   Reverse Depends: kcm (>= 3.9.0.0-0k0.9)
45   Reverse Depends: libkysdk-security (1.2.0.9-0k7.31)
46 libkysdk-security
47   Reverse Depends: kcm (>= 3.9.0.0-0k0.9)
48 libkysset
49 miraclecast
50 ukui-desktop-environment-core
51   Reverse Depends: ukui-desktop-environment (= 2.0.3)
52 ukui-desktop-environment
53 ukui-media
54 kylin-update-frontend
55   Reverse Depends: kcm (>= 3.9.0.0-0k0.9)
56   Reverse Depends: kylin-system-updater (2.0.5.16-0k8.31)
57 preinstalled-apps

```

注意: kylin-system-updater中有反向依赖 kylin-software-properties

status文件内容如下:

```
1 Package: kylin-software-properties
2 Status: install ok installed
3 Priority: optional
4 Section: universe/admin
5 Installed-Size: 683
6 Maintainer: kylin <kylin@kylinos.cn>
7 Architecture: amd64
8 Version: 0.0.1.1-0k5.19
9 Depends: libc6 (>= 2.14), libgcc-s1 (>= 3.0), libqt5core5a (>= 5.12.2),
libqt5dbus5 (>= 5.0.2), libqt5network5 (>= 5.0.2), libssl1.1 (>= 1.1.1),
libstdc++6 (>= 5.2), kylin-daq (>= 1.2.9.1-0k2.13)
10 Conffiles:
11 /etc/cron.d/timermanager 0d1c6acf8e7874e960f4aeceaeda32e7
12 /etc/dbus-1/system.d/com.kylin.software.properties.conf
49a12591ff6464cba04f832a234f1212
13 Description: kylin soft source manager
14
15 Package: kylin-system-updater
16 Status: install ok installed
17 Priority: optional
18 Section: misc
19 Installed-Size: 1212
20 Maintainer: Kylin Development Team <cp@kylinos.cn>
21 Architecture: all
22 Version: 2.0.5.16-0k8.44update1
23 Depends: policykit-1, python3-dbus, python3-psutil, python3-gi (>= 3.8),
python3-yaml, aptdaemon (>= 1.1.1+bzr982-0kylin32.3k6.7), python3-distro-
info, python3-apscheduler, python3-cryptography, sqlite3, kylin-software-
properties (>= 0.0.1.1-0k3.0), kylin-backup-server (>= 1.0.0.0-0k0.16),
kylin-update-frontend, kylin-installer (>= 1.1.86.1-0k0.7), kylin-
background-upgrade (>= 1.3.2.6)
24 Recommends: python3-launchpadlib
25 Suggests: gir1.2-dbusmenu-glib-0.4, gir1.2-unity-5.0
26 Conffiles:
27 /etc/apt/apt.conf.d/30kylin-system-updater a2316b7cf4ea895d2be514ce031d6316
28 /etc/dbus-1/conf/com.kylin.UpgradeStrategies.limit
90f1f620464bcf26f4e0cbe0cb2961e8
29 /etc/dbus-1/conf/com.kylin.UpgradeStrategies.limit.verify
0a89e7866c63a8937ddb4c14e3f056c4
30 /etc/dbus-1/conf/com.kylin.systemupgrade.limit
90f1f620464bcf26f4e0cbe0cb2961e8
31 /etc/dbus-1/conf/com.kylin.systemupgrade.limit.verify
0a89e7866c63a8937ddb4c14e3f056c4
32 /etc/dbus-1/system.d/com.kylin.UpgradeStrategies.conf
ee6a53d10508262d80d0958b2a679410
33 /etc/dbus-1/system.d/com.kylin.systemupgrade.conf
10c82d5f7082d7b9d1ac325b838970c2
34 /etc/kylin-config/basic/com.kylin.UpgradeStrategies.yaml
dda725cde07e282b1fd8202d9a70ded8
35 /etc/kylin-config/basic/com.kylin.systemupgrade.yaml
f3a3bfd242e26a15307ba1f0bc0a3f16
36 /etc/kylin-system-updater/apt_private.conf 90afec35630ab91ca732cbc92d891f41
37 /etc/kylin-system-updater/apt_standard.conf
c5a7f1f0cca99ce96d7c5681fec7e3c4
38 /etc/kylin-version/kylin-system-version.conf
2c9cc0a036ef33aae84db69642005925
39 /etc/logrotate.d/kylin-system-updater 60c4cab34ce58082f6ea5449bfb10b24
```

```
40 /etc/logrotate.d/kylin-unattended-upgrade 0c29fc067eb673c5715d5e9a6d1b571d
41 Description: dbus daemon that manages apt updates.
42 dbus daemon that manages apt updates. Provides DBUS interfaces to UKCC.
43 Homepage: https://www.kylinos.cn
```

可以发现kylin-system-updater中有反依赖 kylin-software-properties，但是status文件中kylin-software-properties的Depends并不存在kylin-system-updater。

解答：

apt-rdepends本质上是个perl脚本，其会通过AptPkg库获取依赖关系（AptPkg 是 Debian/Ubuntu 等 Linux 发行版中用于管理软件包的核心库（C++ 实现），提供了处理 .deb 文件、依赖解析、包元数据读取等底层功能。它是 APT（Advanced Package Tool）的底层支持库，负责解析包索引、处理依赖关系、管理本地包缓存等操作）。

AptPkg库 的所有依赖关系数据均来自 本地 APT 缓存，其核心存储路径为：

/var/lib/apt/lists/：存储从软件源下载的包元数据（如依赖关系、版本信息）。

/var/lib/dpkg/status：记录已安装包的状态和详细信息。

因此，在/var/lib/apt/lists/archive.kylinos.cn_kylin_KYLIN-ALL_dists_10.1_main_binary-amd64_Packages 文件中发现以下内容：

```
1 Package: kylin-software-properties
2 Architecture: amd64
3 Version: 0.0.1.1-0k4.11
4 Priority: optional
5 Section: universe/admin
6 Maintainer: kylin <kylin@kylinos.cn>
7 Installed-Size: 644
8 Depends: libc6 (>= 2.14), libgcc-s1 (>= 3.0), libqt5core5a (>= 5.12.2),
libqt5dbus5 (>= 5.0.2), libqt5network5 (>= 5.0.2), libssl1.1 (>= 1.1.1),
libstdc++6 (>= 5.2), kylin-system-updater (>= 2.0.5.16-0k3.22), kylin-daq
(>= 1.2.9.1-0k2.8)
9 Filename: pool/main/k/kylin-software-properties/kylin-software-
properties_0.0.1.1-0k4.11_amd64.deb
10 Size: 166554
11 MD5sum: 1742f1d9c44138252bda1dd78f0da187
12 SHA1: 32280a09330ae8e3606809494e3997299f1f22c5
13 SHA256: df0c74b4a091e8f4de2b83feb38e47a8f0b4efa69a2dd6eebb294c6ed8f7d669
14 SHA512:
800331bb39bb5c79273ebfb207bf0319be9bb0194ea5e0c4910fed3f308e3fd15b8485637b62
9b6aedf733556d134368596791f2fbf0ebcf116d8f3419e1ca69
15 Description: kylin soft source manager
16 cert_subject_ou: DS120G321040301
17 cert_subject_o: 麒麟软件有限公司
18 cert_subject_cn: 麒麟软件有限公司
19
```

所以apt-rdepends 除了本地安装包信息（/var/lib/dpkg/status）外，还会受到APT源的影响（/var/lib/apt/lists/）。

因此，执行apt-rdepends --state-follow=Installed --state-show=Installed -r {pkg} 获取到的反向依赖关系会比直接从/var/lib/dpkg/status 读出来的更复杂。

实际上，只考虑系统层面的影响域，读取/var/lib/dpkg/status 就足够了。

3.5 组件和系统功能模块如何应对？

前面几个问题，我们弄明白了影响域是什么以及如何获取影响域关系，但是得出的结果还是二进制包名。如果我们只给测试同事二进制包名的信息，测试同事也无从下手。由此，我们还需要搞清楚二进制包与功能组件的关系。

测试部门已经给出了《桌面操作系统模块划分》，在线表格链接如下：

V11: <https://docs.qq.com/sheet/DSkjwZVZEVPVHNX?tab=000001>

V10-SP1: <https://docs.qq.com/sheet/DQ0lRQkhXZXd5bmt2?tab=3j1ht6>

模块划分表内容如图所示：

一级模块	二级模块	三级模块	四级模块	五级模块	对应的关键软件包名（源码包）
	系统安装	本地安装			kylin-os-installer casper initramfs-tools
		GHOST安装			kylin-os-installer
		网络安装			kylin-os-installer
		KVM安装			
		应用预装			preinstalled-apps
	系统激活	二级模块根目录			kylin-activation、libkylin-activation
		产品密钥激活			kylin-activation、libkylin-activation
		二维码激活			kylin-activation、libkylin-activation
		ukey激活			kylin-activation、libkylin-activation、kylin-a
		kms激活			kylin-activation、libkylin-activation
		授权文件导入			kylin-activation、libkylin-activation
	系统规格审查	组件清单			无
		系统缺省配置	基础开机服务		systemd
			基础活跃进程		base-files
		标准符合性	字符编码规范		
			posix标准符合性		
			FHS标准符合性		

模块划分表中有 系统模块-源码包名 的对应关系。

由此可得：

- 1、在kybuilder平台的版本对比中，可以得到 变更软件包列表。
- 2、在status文件中，可以梳理出来 影响域图谱 和 二进制包与源码包的对应关系。
- 3、《桌面操作系统模块划分》表格可以得到 系统模块-源码包 的对应关系

数据处理逻辑如下：

变更包列表 -> 变更包的影响域 -> 受影响二进制包对应的源码包 -> 受影响源码包对应的系统模块

因此，当我们知道变更包列表和基础系统时，我们就能得出哪些系统模块收到影响，将影响模块信息反馈给测试，测试就能针对模块安排测试计划。

四、工具使用

根据需求，已开发单包影响域分析工具，使用说明如下：

使用示例：python3 main.py -id 71542 -t aapt,apt -f test.xlsx -r -m V11.csv

参数：

'-d'('--mount_dir')

必填项(与-id 冲突)

说明: 获取离线status文件，根据参数读取"{mount_dir} + var/lib/dpkg/status"，如果是当前系统，则直接 -d / 即可。

'-id'('--kybuild-id')

必填项(与-d 冲突)

说明: 基础版本的kybuild_id，根据参数从kybuilder平台下载iso的status文件。

'-t'('--target_package')

选填项

说明：目标包名（多个用逗号分隔）。

'-f'('--excel_file')

选填项

说明: 在kybuilder平台https://builder.kylin.com/kybuilder/build/view/{build_id} -> 修改记录 -> 与当前版本对比 -> 导出 即可得到系统差异的excel文件。

如果指定该参数，则会读取该excel文件的包名传入target_package，并且新增<影响域>子表。

如果-t 和 -f 都未指定，则会生成系统全量的关系数据。

注意： excel_file 只能是.xlsx 格式(.xls 格式目前pandas库已经不支持)。

'-g'('--graph_switch')

选填项

说明：生成依赖关系图。

'-r'('--reverse_dep')

选填项

说明：获取影响域关系，默认获取依赖关系。

'-m'('--module-file')

选填项

说明：指定模块信息CSV文件（V11.csv 或者 V10.csv），用于匹配影响域对应的模块

注意：-m 参数需要和 -r 参数一起使用

影响域分析结果会输出到：受影响的模块信息.csv，并且在info.log 中记录了详细的分析过程。

如果指定了-f 参数，则目标excel文件中也会新增一个子表记录影响域结果。