.Deb

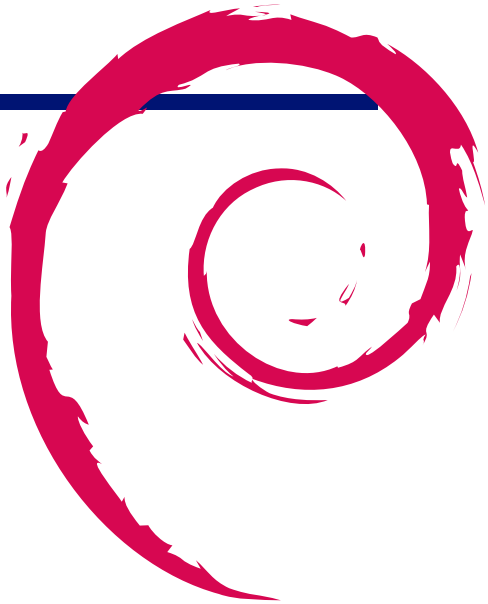
銀河系唯一のDebian専門誌

2017 年 12 月 16 日

gcc pie

デビアン勉強会

目次



1 最近の Debian 関連のミーティング報告

杉本 典充

1.1 第 157 回東京エリア Debian 勉強会

2017 年 11 月 18 日 (土) に第 157 回東京エリア Debian 勉強会を開催しました。会場は横浜にある株式会社ジェイエスピーさんをお借りして行いました。参加者は 14 名とゲストでお呼びした Debian Project Leader である Chris Lamb さんでした。

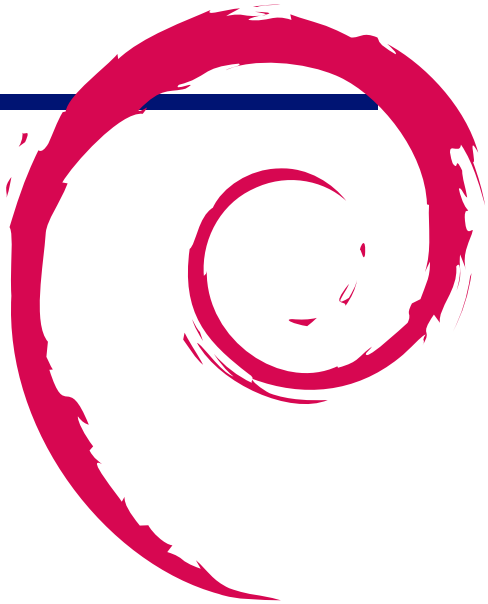
海外の方が参加しているため、勉強会は英語で行われました。自己紹介、自分がやっていること、DPL との会話を英語ですべて行うのは筆者にとってなかなか大変でした。

Chris Lamb さんからは、Debian JP と Debian JP のメンバーが Debian の普及活動を行っていることに理解を示しました。しかし、活動報告やブログなどの情報発信が英語で提供されていないため海外から見ると活動がまったくわからないことを指摘しました。[*1]

セミナーでは、やまねひできさんによる「Rethinking Debian release」を発表しました。”Debian は古い”という言われるゆえんを分析し、Early Majority 向けに適したバージョンをリリースすることで利用ユーザが増えるのではないかと考えました。具体案としてリリースサイクルが短い”Fresh”というバージョンをつくった場合の良い点、悪い点を検討しました。この提案は本発表が初めて公開の場であり、今後 Debian Developer やリリースチームへの展開が進み、議論されることを望みます。

また、前日には Chris Lamb さんの歓迎会を行い、日本とイギリスの料理や文化について語り合いました。

[*1] イベント後、knok さんが英語で勉強会の報告ブログを書いています。 http://blog.daionet.gr.jp/knok-e/2017/11/20/debian-seminar-in-yokohama-20171118/



2 事前課題

杉本 典充

今回の事前課題は以下です:

1. Hack Time は何をしますか
2. OSC のイベント展示で debian の宣伝によさそうなことを 1 つ挙げてください

この課題に対して提出いただいた内容は以下です。

2.1 koedoyoshida

1. 未定
2. メリットの分かりやすいもの、例えばインフォグラフィックスの大きな掲示

2.2 yyatsuo

1. タミル語入力について調べる
2. (回答なし)

2.3 yy_y_ja_jp

1. パッケージ更新
2. (回答なし)

2.4 dictoss

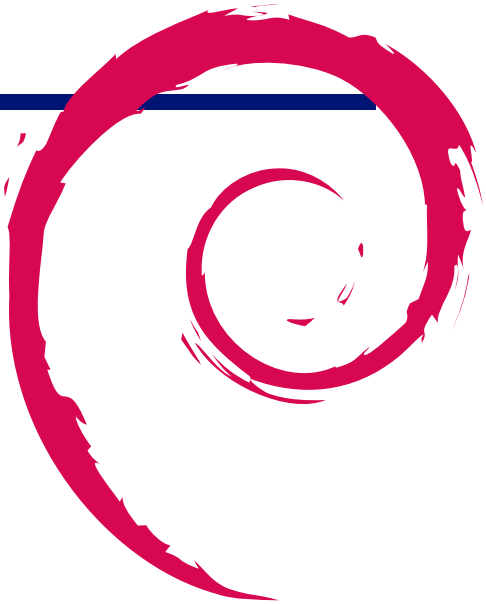
1. ASLR について調べる
2. debian の採用事例集

2.5 Soramichi Akiyama

1. 自作 Slack クライアントの Linux 版のバグ対応（GUI まわり）
2. (回答なし)

2.6 大神祐真

1. 今さらながら、自作ソフトウェアの stretch でのビルド確認などをしようと思っています。
2. あんどきゅめんてっどでびあんバックナンバー展示・頒布



3 gcc の pie オプションと debian における状況について

杉本 典充

3.1 はじめに

Debian 9 のリリースノートに「5.1.5. 実行ファイルはデフォルトで PIE (position independent executables) が有効でコンパイルされています」という記述があります。[*2]

Debian 9 stretch で提供している実行ファイルを file コマンドで確認すると共有オブジェクトと認識しています。[*3]

```
# cat /etc/debian_version
9.1

# file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=3c233e12c466a83aa9b2094b07dbfaa5bd10eccd, stripped
```

この PIE という機能は何なのか気になったため、調べてみました。

3.2 PIE について

3.2.1 PIE とは

PIE（Position Independent Executable、”位置独立実行形式”）とは、PIC（Position Independent Code、”位置独立コード”）のみのオブジェクトファイルで構成した実行ファイルのことをいいます。

PIC であるオブジェクトファイルは、機械語を相対アドレスで記述します。主に共有ライブラリに含んだオブジェクトファイルは”-fPIC”オプションを適用してコンパイルすることで PIC にするのが一般的です[*4]。PIE な実行ファイルは、実行ファイルが保持するオブジェクトファイルの機械語部分（= ELF 形式の.text 領域）を PIC にしたものです。実行ファイル、ライブラリのコードをすべて PIC にすることで、機械語は仮想アドレスのどの番地に配置されても実行できるようになります。

（非 PIE な）今までの実行ファイルの場合、実行ファイルや共有ライブラリの機械語を仮想アドレスのどの位置に配置するかはリンク時に決定します。そのため、実行時の仮想アドレスは毎回同じ位置に同じデータが配置されます。

PIE な実行ファイルを実行したとき、ダイナミックリンカー（ld.so や ld-linux.so）は相対アドレスで記述された機械語を絶対アドレスに変換して仮想アドレス空間に配置する前処理を実行し、そのあとプログラムを実行します。このアドレス変換処理は、実行ファイル内の機械語と共有ライブラリ内の機械語の両方が処理対象になります。

[*2] https://www.debian.org/releases/stretch/amd64/release-notes/ch-information.ja.html#pie-is-now-default
[*3] Debian 8 jessie では「ELF 64-bit LSB executable」という表示になります。
[*4] 非 PIC でも共有ライブラリとして動きますが、メモリ効率、実行速度が劣ります。

3.2.2 PIE と非 PIE のアドレス配置の差異の確認

以下のコードで確認します。単に、foo() の関数アドレスを print するだけのものです。

```
$ cat test.c
#include <stdio.h>

void foo()
{
  printf(``IN foo()\n'');
  return;
}

int main()
{
  void (*f)() = foo;
  f();

  printf(``0x%x\n'', f);
  return 0;
}
```

非 PIE でコンパイルして実行します。（Debian 8 jessie の gcc-4.9.2 を利用）実行時の foo() のアドレスが毎回同じであることがわかります。

```
$ gcc test.c
$ ./a.out
IN foo()
0x400546
$ ./a.out
IN foo()
0x400546
$ ./a.out
IN foo()
0x400546
```

PIE でコンパイルして実行します。（Debian 9 stretch の gcc-6.3.0 を利用）実行時の foo() のアドレスが毎回異なることがわかります。

```
$ gcc test.c
$ ./a.out
IN foo()
0xae27f6f0
$ ./a.out
IN foo()
0xbb1466f0
$ ./a.out
IN foo()
0xbe86b6f0
$ ./a.out
IN foo()
0x37f456f0
```

3.2.3 PIE の長所と短所

PIE の長所として、アドレス変換処理を行うことで実行プログラムを仮想アドレスに配置した結果が実行する度にランダムになるため、脆弱性があるプログラムは特定のコードを実行されにくくなります。そのため、セキュリティが向上します。

PIE の短所として、PIE な実行ファイルの実行にはアドレス変換処理そのもののオーバーヘッドがあり、実行速度は非 PIE な実行ファイルに比べて遅くなります。

しかし、実行速度が遅い短所は gcc-5.0 において Intel による開発の成果が取り込まれ、改善されてきた経緯があります[*5]。

3.2.4 PIE の実行ファイルを生成する

PIE の実行ファイルを生成するには以下の手順でコンパイル、リンクする必要があります。

- gcc でソースコードをコンパイルする → ”-fPIE”オプションを付与する
- gcc でオブジェクトファイルをリンクする → ”-pie”オプションを付与する

[*5] https://software.intel.com/en-us/blogs/2014/12/26/new-optimizations-for-x86-in-upcoming-gcc-50-32bit-pic-mode

- PIE な実行ファイルを実行する → 対応しているダイナミックローダーが必要

そのため、PIE の実行ファイルを生成し実行するには、コンパイラ（gcc-3.4 以降）、リンカー（ld コマンド。binutils-2.15 以降)、ダイナミックリンカー（glibc に含まれる ld.so、ld-linux.so など）の対応が必要です。

また、gdb は gdb-7.1 から PIE の実行ファイルのデバッグが可能になっています。[*6]

3.3 PIE の採用状況

PIE はセキュリティを高める機能のため、セキュリティに力を入れているディストリビューションで採用が始まりました。

- OpenBSD
  - 「OpenBSD's Position Independent Executable (PIE) Implementation」 `http://www.openbsd.org/papers/nycbsdcon08-pie/mgp00001.html`
  - 「Converting OpenBSD to PIE」`https://www.openbsd.org/papers/asiabsdcon2015-pie-slides.pdf`
  - OpenBSD 5.3（2013-05-01 にリリース）で PIE をデフォルトの実行ファイルになった。
- Fedora
  - 「Changes/Harden All Packages」 `https://fedoraproject.org/wiki/Changes/Harden_All_Packages`
  - Fedora 23（2015-11-03 にリリース）で PIE をデフォルトの実行ファイルになった。
- Ubuntu
  - 「GCC hardening for 16.10」 `https://wiki.ubuntu.com/SecurityTeam/PIE`
  - Ubuntu 16.10（2016-10-13 にリリース）で PIE をデフォルトの実行ファイルになった。
- Debian
  - リリースノート `https://www.debian.org/releases/stretch/amd64/release-notes/ch-information.ja.html#pie-is-now-default`
  - Debian 9（2017-06-16）で PIE をデフォルトの実行ファイルになった。
- HardenedBSD
  - FreeBSD の fork。`https://hardenedbsd.org/`
  - 11-STABLE をベースにしたバージョンをリリースしている。
- Android
  - Android 5.0 からは、PIE な実行ファイルのみが実行可能。 2014-10-17 にリリース。

3.4 Debian における PIE の状況

3.4.1 PIE に関する情報提供

Debian における、セキュリティの強化は以下の”Hardening”のページに記載があります。PIE についての記述もこのページにあります。

`https://wiki.debian.org/Hardening`

過去の debian では、debian パッケージのビルド時の環境変数”DEB_BUILD_HARDENING=1”を指定すると”-fPIE -pie”オプションが指定されるようになり、PIE な実行ファイルを生成することができます。

Debian 9 で提供する実行ファイルのデフォルト形式を PIE で提供することに対する議論は、以下のページとメーリングリストで見ることができます。

[*6] `https://lwn.net/Articles/379511/`

- https://wiki.debian.org/Hardening/PIEByDefaultTransition
- 「Porter roll call for Debian Stretch」https://lists.debian.org/debian-devel/2016/08/msg00324.html

また、Lintian には「hardening-no-pie」という warning があり、パッケージのビルドログに出力されます[*7]。

3.4.2 gcc パッケージ

Debian 9 の gcc パッケージは gcc-6.3.0 を採用されており、”gcc -V”を実行して configure オプションを確認すると、”–enable-default-pie”を指定しています。

```
$ gcc -v 2>&1 | grep pie
  Configured with: ../src/configure -v --with-pkgversion='Debian 6.3.0-18'
  --with-bugurl=file:///usr/share/doc/gcc-6/README.Bugs
  --enable-languages=c,ada,c++,java,go,d,fortran,objc,obj-c++ --prefix=/usr
  --program-suffix=-6 --program-prefix=x86_64-linux-gnu- --enable-shared
  --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext
  --enable-threads=posix --libdir=/usr/lib --enable-nls --with-sysroot=/
  --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes
  --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify
  --enable-libmpx --enable-plugin
  --enable-default-pie
  --with-system-zlib --disable-browser-plugin  --enable-java-awt=gtk --enable-gtk-cairo
  --with-java-home=/usr/lib/jvm/java-1.5.0-gcj-6-amd64/jre --enable-java-home
  --with-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-6-amd64
  --with-jvm-jar-dir=/usr/lib/jvm-exports/java-1.5.0-gcj-6-amd64
  --with-arch-directory=amd64 --with-ecj-jar=/usr/share/java/eclipse-ecj.jar
  --with-target-system-zlib --enable-objc-gc=auto --enable-multiarch
  --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32
  --enable-multilib --with-tune=generic --enable-checking=release
  --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
```

gcc のマニュアルを確認すると、「Turn on -fPIE and -pie by default. 」と記述があります https://gcc.gnu.org/install/configure.html。そのため、debian 9 以降の gcc を使ってアプリケーションをコンパイルすると、デフォルトで PIE な実行ファイルが生成されます。

なお、PIE をデフォルトで有効にする CPU アーキテクチャはホワイトリストで gcc-6 ソースパッケージに書かれています。[*8]

```
  $ apt-get source gcc-6
  $ cd gcc-6-6.3.0/debian
  $ less rules.defs
  (snip)
  # pie by default --------------------
  with_pie :=
  ifeq ($(distribution),Debian)
    ifeq (,$(filter $(distrelease),wheezy squeeze jessie))
      pie_archs = amd64 arm64 armel armhf i386 mips mipsel mips64el \
                  ppc64el s390x sparc sparc64 kfreebsd-amd64 kfreebsd-i386
    endif
  else ifeq ($(distribution),Ubuntu)
    ifeq (,$(filter $(distrelease),lucid precise trusty utopic vivid wily))
      pie_archs = s390x
    endif
    ifeq (,$(filter $(distrelease),lucid precise trusty utopic vivid wily xenial))
      pie_archs += amd64 ppc64el
    endif
  endif
  ifneq (,$(filter $(DEB_TARGET_ARCH),$(pie_archs)))
    with_pie := yes
  endif
  (snip)
```

3.4.3 PIE 形式にしたくない場合

gcc のリンクオプション”-no-pie”を指定すると、非 PIE の実行ファイルを生成できます。

[*7] https://lintian.debian.org/tags/hardening-no-pie.html

[*8] hppa と m68k では PIE の実行ファイルは動作しないようです。https://wiki.debian.org/Hardening

```
$ vi test.c
int main()
{
  return 0;
}

$ gcc -no-pie test.c
$ file ./a.out
./a.out: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=032331c152d5bab1c160a884967d9cd9507a73e6, not stripped
```

3.5 終わりに

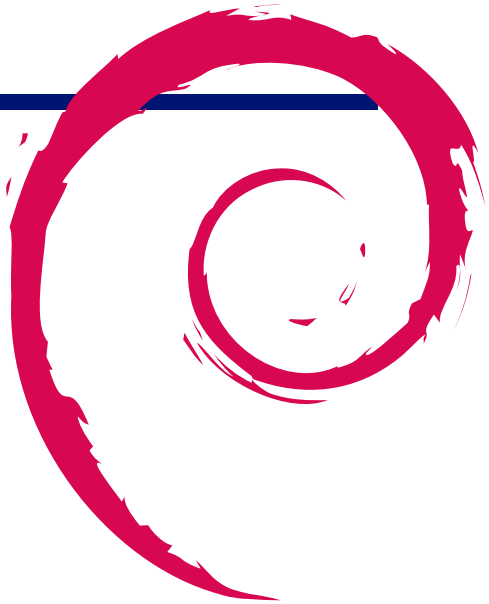
PIE について調べてみました。PIE は PIC の概念の延長のため、PIC も勉強になりました。

今回の PIE に似た技術として、実行プログラムのスタック領域やヒープ領域もランダム化して実行する ASLR（Address Space Layout Randomization）という機能もあるとのことです。

プログラムがどのように実行されるのかを知っておくのは有意義だと思います。

3.6 参考文献

- Debian Wiki Hardening
  - https://wiki.debian.org/Hardening
- Debian Wiki Hardening PIEByDefaultTransition
  - https://wiki.debian.org/Hardening/PIEByDefaultTransition
- ももいろテクノロジー ELF 実行ファイルのメモリ配置はどのように決まるのか
  - http://inaz2.hatenablog.com/entry/2014/07/27/205913
- Red Hat Security Blog Position Independent Executables (PIE)
  - https://access.redhat.com/blogs/766093/posts/1975793
- OpenBSD's Position Independent Executable (PIE) Implementation
  - http://www.openbsd.org/papers/nycbsdcon08-pie/mgp00001.html
- New optimizations for X86 in upcoming GCC 5.0: PIC in 32 bit mode.
  - https://software.intel.com/en-us/blogs/2014/12/26/new-optimizations-for-x86-in-upcoming-gcc-50-32bit-pic-mode



4 メモ

Debian 勉強会資料

2017 年 12 月 16 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）