Computer Security Symposium 2018
22 - 25 October 2018

Linuxディストリビューション間でのELFバイナリにおけるセキュリティ対策機構の適用状況の比較

齋藤 孝道[1]　菅原 捷汰[2]　横山 雅展[2]　鈴木 嵩人[1]　石渡 聖也[1]　須崎 有康[3]

概要：Linux ディストリビューションのセキュリティ管理を考慮する上で，ELF バイナリにおけるセキュリティ対策機構の適用状況，32 ビットと 64 ビットでの違い，ディストリビューション間の違い，世代間の変化は重要な知識である．しかし，上記の分析に必要となる ELF バイナリのビルドされた環境，コンパイルオプションといった情報は ELF バイナリ本体のみからは得ることができない．そこで，ELF バイナリ本体とそのパッケージの双方に着目し，3 種の Linux ディストリビューション各 4 世代（32 ビット，64 ビットを含む）において静的解析を行った．本論文では，静的解析から得られた結果の比較と考察を行った．特に，比較については同一ディストリビューションのバージョン間での比較に加え，新たに異なるディストリビューション間においてもそれぞれに共通して存在するバイナリに着目した比較も行った．

キーワード：セキュリティ対策機構，共通バイナリの比較

Comparison of Application Status of Security Mechanisms in ELF Binaries among Linux Distributions

Takamichi Saito[1]　Shota Sugawara[2]　Masahiro Yokoyama[2]　Takahito Suzuki[1]
Seiya Ishiwatari[1]　Kuniyasu Suzaki[3]

Abstract: In considering the security management of Linux distributions, the application status of security mechanisms in ELF binaries, the difference between 32-bit and 64-bit, the difference between distributions, and the change between generations are important knowledge. However, information such as the build environment, compilation options, etc. required for the above analysis can not be obtained only from the ELF binary itself. Therefore, focusing on both the ELF binary body and its package, we carried out static analysis in each of the three types of Linux distributions (including 32-bit and 64-bit). In this paper, we compared and considered the results obtained from the static analysis. In particular, we compared comparisons between versions of the same distribution, as well as comparisons focusing on binaries that are common among different distributions as well.

Keywords: Security Mechanisms, Comparison of Common Binary

1. はじめに

CWE-119[1] に分類されるメモリ破壊脆弱性を悪用する攻撃（以降，メモリ破壊攻撃と呼ぶ）に対するセキュリティ対策機構が様々に考案されてきた．既に一部の対策機構は OS や主要なコンパイラに標準で組み込まれ，容易に利用できる．

しかし，メモリ破壊攻撃の多様化も進んでおり，ROP（Return Oriented Programming）に代表されるコード再利用攻撃は，先行研究 [2] や [3] のように高度な手法に派生しており，既存の対策機構を回避する攻撃が次々に登場し

[1] 明治大学
Meiji University
[2] 明治大学大学院
Graduate School of Meiji University
[3] 国立研究開発法人産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

ている．多様化するメモリ破壊攻撃を防御，緩和する観点では，複数の対策機構を組み合わせて適用することが望ましいとされている．

しかしながら，既存研究 [4] において，CentOS，Ubuntu，openSUSE の 3 種類の Linux ディストリビューションの 4 世代の 32bit/64bit に標準で含まれる ELF バイナリを解析し，GCC に組み込まれている対策機構のうち RELRO，SSP，PIE，Automatic Fortification の 4 つについて適用状況の調査が行われた結果，RELRO と PIE は全体的に適用率が低く，安全とは言い切れないバイナリが一定数存在することが示された．

本論文では，3 つの Linux ディストリビューション（CentOS，Ubuntu，openSUSE）の 4 世代の 32/64bit 版に標準で含まれる ELF バイナリを解析し，4 つのセキュリティ対策機構（RELRO，SSP，PIE，Automatic Fortification）について以下の調査を行った．

- セキュリティ対策機構の導入傾向
- 同一ディストリビューションの異なるバージョン間で共通する ELF バイナリに着目した対策機構の適用状況の変化
- 異なるディストリビューション間で共通する ELF バイナリに着目した対策機構の適用状況の比較

既存研究は，Ubuntu のマイナーバージョンである 12.04.5 と 14.04.5 を調査対象としていたので，本論文では 12.04 と 14.04 を調査対象とし，各ディストリビューション間のリリース日のずれが小さくなるようにした．また，本論文では，既存研究で行った同一ディストリビューションのバージョン間での対策機構の適用状況の比較に加えて，各ディストリビューションに共通して存在するバイナリを母集団として，異なるディストリビューション間での対策機構の適用状況の比較も行った．

上記の変更に伴い，本論文での調査対象の CentOS，Ubuntu，openSUSE のバイナリ数は，それぞれ，10,542 個，8,283 個，15,424 個で，総数 34,249 個となった．

本論文の狙いと目的は，バイナリのセキュリティ対策機構の適用状況を明らかにすることで，脆弱性を悪用される可能性があるバイナリを可視化することや母集団を合わせて対策機構の適用状況を比較することで，セキュリティが高いディストリビューションを明確にすることである．

本論文での調査の結果，以下のことがわかった．

(1) ディストリビューションごとに，セキュリティ対策機構の適用状況が違っていた
(2) セキュリティ対策機構が，バイナリに一旦適用されたのち，後の世代で意図的に，非適用・弱化されたケースが散見された
(3) コンパイルオプションは指定されていたが，実際には，機能していないケースがあった

特に，Ubuntu17.04 は調査対象としたすべての対策機構の適用率が約 90%であり，セキュリティが高いことが言える．

2. 調査対象とするセキュリティ対策機構

メモリ破壊攻撃へのセキュリティ対策機構はこれまでに様々なものが提案され，一部のセキュリティ対策機構は OS や，GCC に代表される主要なコンパイラに標準で組み込まれている．本節では，調査対象とする 4 つの対策機構について取り上げる．

- RELRO（RELocation Read-Only）
  ELF バイナリの.got セクションを読み込み専用にするリンカによる対策機構である．RELRO は遅延バインドが有効の場合は Partial RELRO が適用され，仮想アドレス空間内に.got セクションとは別に読み書き可能な.got.plt セクションが生成される．遅延バインドが無効の場合は，Full RELRO が適用され，.got.plt セクションは生成されず，データセグメント以外読み込み専用となる．
- SSP（Stack Smashing Protector）
  コンパイル時に，Stack-based Buffer Overflow 攻撃を検知する検査コードを，適用対象のプログラムに挿入する対策機構である．プログラムの実行時に Stack-based Buffer Overflow 攻撃を検知した場合は，プログラムの実行を停止する．
- PIE（Position Independent Executable）
  カーネルが提供するメモリランダマイズ化の ASLR（Address Space Layout Randomization）をテキスト領域で実現する対策機構である．実行バイナリのアドレス参照を相対アドレスにすることで，その実行バイナリが仮想アドレス空間のどこに配置されても正常に実行できるようにする．
- Automatic Fortification
  コンパイル時に，Buffer Overflow 脆弱性の原因となりうるライブラリ関数を書き込み先のバッファの境界検査を行う安全な代替関数に置換する対策機構である．置換された関数によって，Buffer Overflow を検出した場合，プログラムの実行を停止する．Automatic Fortification を有効にした時の関数の置換は，書き込み先のバッファサイズおよび書き込むデータサイズによって変化する [8][9].

3. 調査方法

3.1 調査対象のディストリビューションとバージョン

本論文で調査の対象とする Linux ディストリビューションとそのバージョンを**表 1** に示す．Red Hat 系から CentOS，Slackware 系から openSUSE，Debian 系から Ubuntu を選定した．

各ディストリビューションのバージョンの選定は，異な

表 1　調査対象のディストリビューションとバージョン

ディストリビューション	バージョン	bit 数	リリース日
Red Hat 系 CentOS	6.3	32/64bit	2012-07-09
	6.5	32/64bit	2013-12-01
	6.6	32/64bit	2014-10-28
	7.4	64bit	2017-09-14
Debian 系 Ubuntu	12.04	32/64bit	2012-04-26
	13.04	32/64bit	2013-04-25
	14.04	32/64bit	2014-04-17
	17.04	64bit	2017-04-13
Slackware 系 openSUSE	12.2	32/64bit	2012-09-05
	13.1	32/64bit	2013-11-19
	13.2	32/64bit	2014-11-04
	42.3	64bit	2017-07-26

るディストリビューション間の比較および最新のバージョンの対策機構の適用状況の調査の観点から行った．異なるディストリビューション間で比較を行うために，連続した 3 年間（2012 年，2013 年，2014 年）にリリースされた 32/64bit 版を各ディストリビューションにつき 3 バージョン選定した．

さらに，比較的新しいバージョンの対策機構の適用状況を調査するために各ディストリビューションにつき，2017 年にリリースされたバージョンを 1 つ選定した．Ubuntu17.04 には 32bit 版も存在するが，CentOS と openSUSE には存在しない．調査条件を可能な限り合わせるために，今回は，64bit 版のみを選定した．調査対象の各ディストリビューションはデスクトップバージョンである．

3.2　対策機構の適用状況の調査手法

対策機構の適用状況の調査手法は各ディストリビューションの root ユーザにおける環境変数 PATH が通っているディレクトリ内のバイナリおよびシンボリックリンクが指す先のバイナリに対して，trapkit[10] の checksec を参考に作成したスクリプトを実行するというものである．スクリプトの実行結果から，ディストリビューションおよびバージョンごとに対策機構の適用状況を比較する．

3.2.1　RELRO の調査方法

RELRO の有効および無効の分類は対象のバイナリの GNU_RELRO セグメントの有無で行っている．GNU_RELRO セグメントが存在すれば，RELRO が有効であり，さらに対象のバイナリの.dynamic セクションのエントリタイプに BIND_NOW が存在すれば Full RELRO，存在しなければ Partial RELRO として分類する．

3.2.2　SSP の調査方法

SSP の有効および無効の分類は，検査コードとして追加される `__stack_chk_fail` 関数の有無で行っている．検査コードはローカル変数に文字配列がない関数や，文字配列が 8 バイト未満である関数では挿入されない．対象のバイナリの全ての関数で検査コードが挿入されていない場合，SSP が有効でコンパイルされていても，本論文の調査では無効に分類する．

3.2.3　PIE の調査方法

PIE の有効および無効の分類は対象のバイナリの ELF ヘッダのタイプが DYN かどうかで行っている．DYN であれば PIE が有効でコンパイルされているので，有効として分類し，それ以外は無効として分類している．

3.2.4　Automatic Fortification の調査方法

Automatic Fortification の有効および無効の分類は，書き込み先のバッファの境界検査を行う安全な代替関数として追加される `_chk` を接尾辞に持つ関数の有無で行っている．Automatic Fortification の安全な代替関数の置換条件を満たしておらず，置換が行われなかったバイナリは，Automatic Fortification が有効としてコンパイルされている場合でも，本論文の調査では無効に分類する．

4.　調査結果

4.1　各対策機構の適用状況の調査結果

各ディストリビューションのバイナリ数を**表 2** に示す．32/64bit のバイナリ数は概ね同数であり，後述する各対策機構の適用状況も 32/64bit でほとんど違いがなかったので，本節の各対策機構の適用状況は，64bit のみを示す．

表 2　調査したバイナリの総数

ディストリビューション	バージョン	バイナリの数 32bit	バイナリの数 64bit
CentOS	6.3	1,499	1,497
	6.5	1,487	1,485
	6.6	1,494	1,492
	7.3	N/A	1,588
Ubuntu	12.04	1,104	1,103
	13.04	1,152	1,152
	14.04	1,161	1,161
	17.04	N/A	1,450
openSUSE	12.2	2,181	2,178
	13.1	2,199	2,199
	13.2	2,226	2,228
	42.3	N/A	2,213

4.1.1　RELRO の適用状況

図 1 は 64bit における RELRO の適用状況である．

CentOS6.3，6.5 および 6.6 では非適用の割合が他のディストリビューションに比べて高いが，最新バージョンである 7.4 で改善されている．Full RELRO が適用されているバイナリの割合は Ubuntu17.04 のみ 90%を上回っていた．

4.1.2　SSP の適用状況

図 2 は 64bit における SSP の適用状況である．

どのディストリビューションにおいても，SSP が適用されているバイナリの割合は非適用に比べて高かった．CentOS7.4，Ubuntu17.04 を除いて，各ディストリビュー

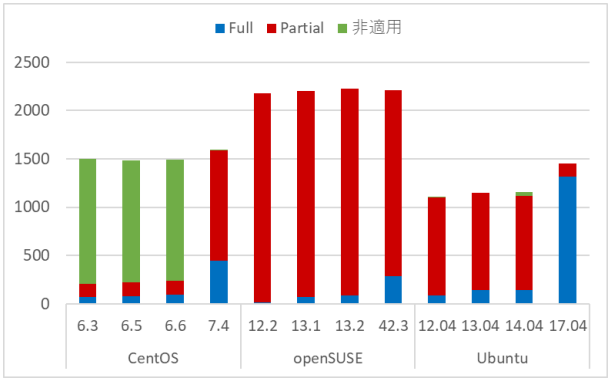

図 1　64bit における RELRO の適用状況

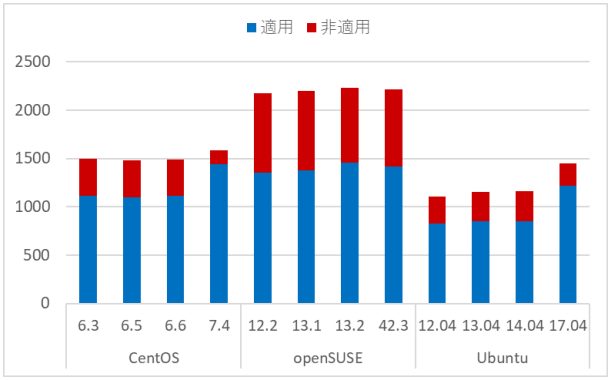

図 2　64bit における SSP の適用状況

ションのバージョン間では適用率が横這いであることがわかる．

4.1.3　PIE の適用状況

図 3 は 64bit における PIE の適用状況である．

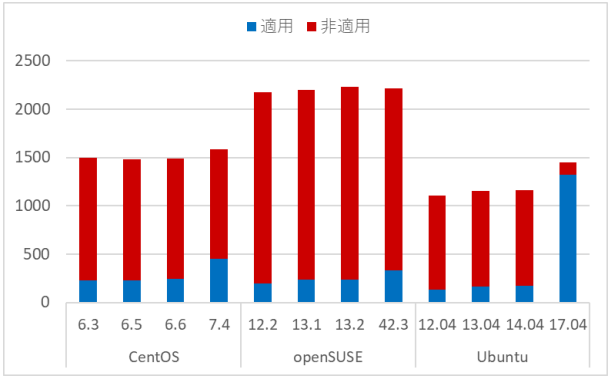

図 3　64bit における PIE の適用状況

Ubuntu17.04 以外のディストリビューション全てで，PIE が適用されているバイナリの割合は非適用に比べて低かった．適用率の変遷は概ね横這いであるが，各ディストリビューションの最新バージョンでは適用率が上昇している傾向にある．

4.1.4　Automatic Fortification の適用状況

図 4 は 64bit における Automatic Fortification の適用状況である．

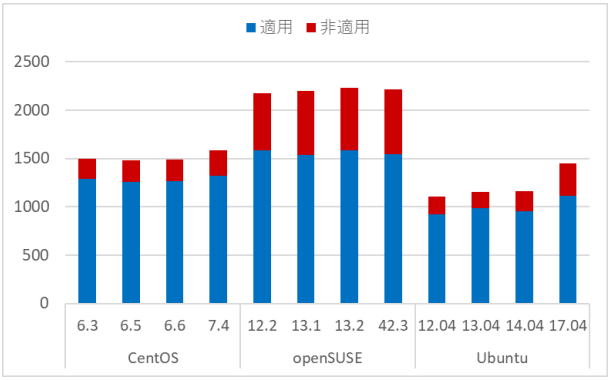

図 4　64bit における Automatic Fortification の適用状況

どのディストリビューションにおいても，Automatic Fortification が適用されているバイナリの割合は非適用に比べて高かった．他の対策機構の適用状況と違い，最新バージョンでの適用率の上昇は見られなかった．

4.2　適用状況の変化

本論文では，各ディストリビューションのバージョン間で共通するバイナリについて対策機構の適用状況の変化についても調査した．以降，非適用から適用または Partial RELRO から Full RELRO への変化を強化，適用から非適用または Full RELRO から Partial RELRO への変化を弱化と呼ぶ．

表 3 に 64bit における「強化/弱化」したバイナリの数を示す．一定数のバイナリが強化されている一方で，弱化されているバイナリも一定数存在した．本論文では Ubuntu12.04 と Ubuntu14.04 間および Ubntu14.04 と Ubuntu17.04 間で弱化したバイナリについて，パッケージから再ビルドを行いコンパイルオプションの有無や対策機構の適用条件に当てはまるか確認した．Ubuntu14.04 と Ubuntu17.04 間で弱化したバイナリの数は SSP で 1 個，Automatic Fortification で 3 個存在した．RELRO と PIE には弱化したバイナリは存在しなかった．

4.2.1　RELRO と Automatic Fortification の弱化

Ubuntu12.04 と Ubuntu14.04 間では，Automatic Fortification が弱化しているバイナリが 40 個存在した．

これら 40 個のうち，33 個のバイナリは，Automatic Fortification が弱化すると同時に，RELRO も弱化していた．

この 33 個のバイナリは，3 つのパッケージ（x11-apps, x11-xfs-utils，x11-xkb-utils）に属し，いずれも，X Window System に関係するものであった．これらのバイナリのビルド時のコンパイルオプションを確認した．その結果，Ubuntu12.04 では，Automatic Fortification と Partial RELRO のオプションが明示的に指定されていたが，Ubuntu14.04 では，そのどちらのオプションも指定されてないことがわかった．また，残りの 7 個のバイナリのうち

表 3　64bit における強化/弱化されたバイナリの数

		強化されたバイナリ数/弱化されたバイナリ数				共通のバイナリ数
ディストリビューション	比較バージョン	RELRO	SSP	PIE	Automatic Fortification	
CentOS	6.3 => 6.5	11/0	6/3	2/0	0/4	1438
	6.3 => 6.6	25/0	7/3	16/0	0/4	1438
	6.5 => 6.6	14/0	1/0	14/0	0/0	1477
Ubuntu	12.04 => 13.04	52/6	0/8	29/6	16/2	1057
	12.04 => 14.04	54/54	4/13	32/6	20/40	1015
	13.04 => 14.04	3/51	5/7	3/0	9/41	1105
openSUSE	12.2 => 13.1	25/1	17/8	25/0	5/12	1915
	12.2 => 13.2	34/1	44/9	24/1	22/13	1817
	13.1 => 13.2	10/0	31/1	2/1	21/1	2075

5 個のバイナリは，1 つのパッケージ（xfonts-utils）に属する．これらのバイナリのビルド時のコンパイルオプションを確認したところ，Ubuntu12.04 でも Ubuntu14.04 でもコンパイルオプションが指定されていなかった．しかし，Ubuntu12.04 では，Automatic Fortification による関数の置換が行われていた．残り 2 個のバイナリは，2 つのパッケージ（evince，nautilus）に属する．これらのバイナリは，Ubuntu12.04 と Ubuntu14.04 のどちらもビルド時のコンパイルオプションが指定されていた．しかし，Ubuntu14.04 においては置換対象の関数のシンボルが存在しなかった．

一方，Ubuntu14.04 から Ubuntu17.04 間で Automatic Fortification が弱化した 3 個のバイナリは 2 つのパッケージ（apt，eog）に属する．これらのバイナリのビルド時のコンパイルオプションを確認したところ，Ubuntu14.04 と Ubuntu17.04 の双方でコンパイルオプションが指定されていた．しかし，Ubuntu14.04 で置換されていた関数が Ubuntu17.04 では存在しなかった．

4.2.2　PIE の弱化

Ubuntu12.04 と Ubuntu14.04 間では，PIE が弱化しているバイナリが 6 個あり，全て dbus パッケージに属していることがわかった．Automatic Fortification のケースと同様に，このパッケージのコンパイルオプションを確認したところ，Ubuntu14.04 では，PIE を有効にするコンパイルオプションが指定されていなかった．この原因を調査したところ，Ubuntu14.04 の dbus パッケージでは PIE を有効にすると正しく動作しないという理由から，明示的に PIE が無効化されていることが分かった [12][13].

4.2.3　SSP の弱化

Ubuntu12.04 と Ubuntu14.04 間で，SSP が弱化したバイナリが 13 個存在した．これら 13 個のうち，2 個のバイナリは，2 つのパッケージ（x11-xfs-utils，x11-xkb-utils）に属する．これらのバイナリのビルド時のコンパイルオプションを確認したところ，Ubuntu12.04 と Ubuntu14.04 のどちらのバージョンでもコンパイルオプションが指定されていなかった．しかし，Ubuntu12.04 ではコンパイラオプションが指定されてないにも関わらず，検査コードが挿入されていた．残りの 11 個のうち 1 個のバイナリは，1 つのパッケージ（intel-gpu-tools）に属する．これらのバイナリのビルド時のコンパイルオプションを確認した．Ubuntu12.04 では，コンパイルオプションが指定されていたが，Ubuntu14.04 ではコンパイルオプションが指定されていなかった．残りの 10 個のバイナリでは，どちらのディストリビューションもコンパイルオプションが指定されていた．10 個のうち 3 個のバイナリは，2 つのパッケージ（colord，gcalctool[*1]）に属する．これら 3 個のバイナリの，ソースコードを確認した．その結果，SSP の適用条件に当てはまらないことがわかった．このことから，SSP のコンパイルオプションを指定しても SSP の検査コードが挿入されなかったことが考えられる．

残りの 7 個のバイナリは，どちらのディストリビューションもコンパイルオプションが指定されており，SSP の適用条件に当てはまるにもかかわらず，SSP の検査コードが挿入されていなかった．これはバグである可能性が高い．

一方，Ubuntu14.04 から Ubuntu17.04 間で SSP が弱化した 1 個のバイナリは 1 つのパッケージ（crda）に属する．このバイナリのビルド時のコンパイルオプションを確認したところ，Ubuntu14.04 と Ubuntu17.04 の双方でコンパイルオプションが明示的に指定されていた．しかし，Ubuntu14.04 で SSP の検査コードが挿入された関数が Ubuntu17.04 では存在しなかった．

4.3　ディストリビューション間の比較

本論文では，バージョン間の比較だけではなく，ディストリビューション間で共通する ELF バイナリを対象に，年代別で対策機構の適用状況の比較を行った．年代別で共通する ELF バイナリの総数を**表 4** に示す．共通のバイナリの種類は，Linux を構成するものや Linux のコマンド，通信に関わるものなどが含まれていた．

ディストリビューション間の比較では，RELRO は Full

[*1] Ubuntu14.04 では gnome-calculator に変更された．

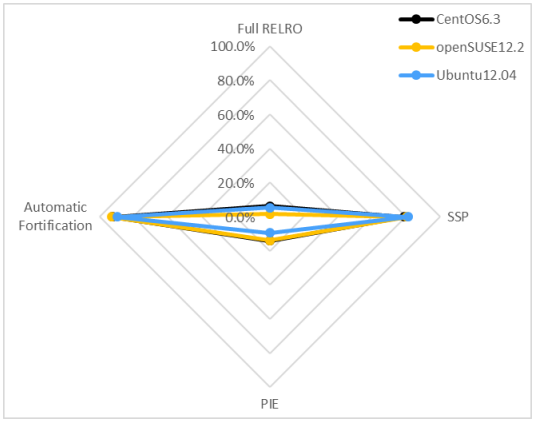

図 5　ディストリ間の共通バイナリの対策機構比較 2012 年

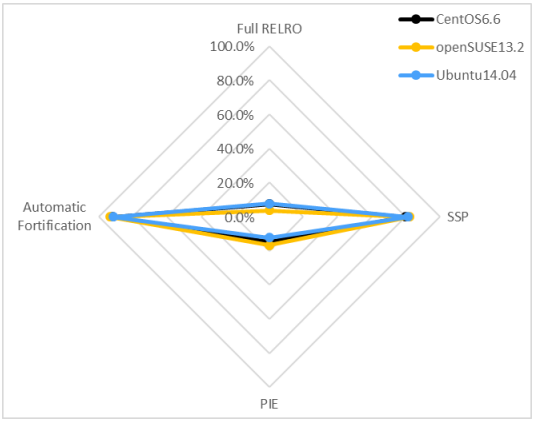

図 7　ディストリ間の共通バイナリの対策機構比較 2014 年

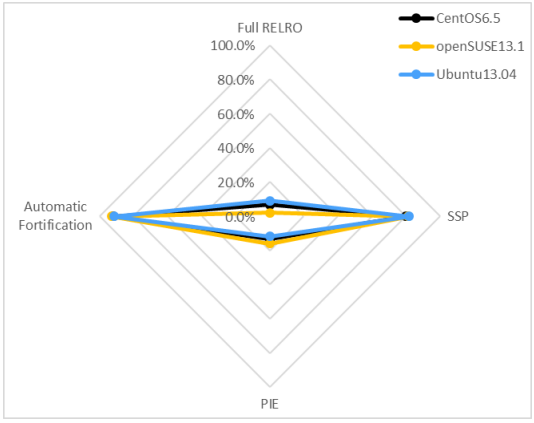

図 6　ディストリ間の共通バイナリの対策機構比較 2013 年

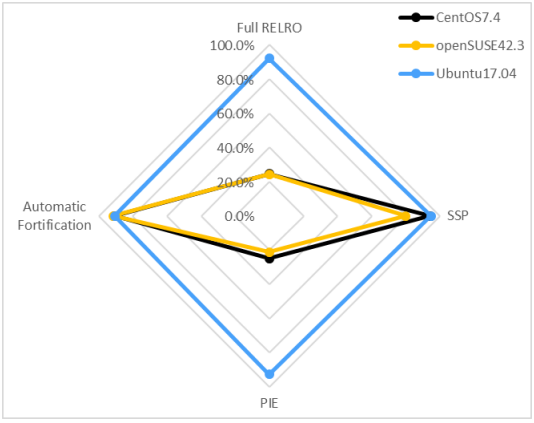

図 8　ディストリ間の共通バイナリの対策機構比較 2017 年

表 4　年代別の共通の ELF バイナリの総数

	年代			
	2012 年	2013 年	2014 年	2017 年
総数	614	574	571	612

RELRO を比較対象とした．また，比較対象のディストリビューションは 64bit である．

図 5，**図 6**，**図 7**，**図 8** はそれぞれ 2012 年，2013 年，2014 年，2017 年のディストリビューションを比較したチャート図である．どの年代を見ても，SSP と Automatic Fortification が適用されている ELF バイナリの割合が高く，Full RELRO や PIE の割合が低かった．

しかし，図 8 を見ると Ubuntu17.04 が他のディストリビューションに比べ，Full RELRO と PIE が適用されている ELF バイナリの割合が高くなっている．文献 [11] によると Ubuntu16.10 からコンパイル時にデフォルトで PIE が適用されるようになっている．このことから，Ubuntu における PIE の重要度が高くなっていることがわかる．以上より，Ubuntu17.04 は他の 2 つのディストリビューションと比べてセキュリティが高いことが言える．

さらに，2017 年にリリースされたディストリビューションに共通するバイナリの内，各対策機構がどのディストリビューションでも適用されていなかったものについて，その原因がソースコードの不備にあるのか調査した．

3 つのディストリビューション全てで Full RELRO が無効のバイナリは 41 個，SSP が無効のバイナリは 26 個，PIE が無効のバイナリは 36 個，Automatic Fortification が無効のバイナリは 49 個であった．

調査は，Ubuntu17.04 において上記のバイナリが属する 43 のパッケージを各対策機構を有効にするコンパイルオプションを指定した上でビルドし，コンパイルエラーが出ないか，本当に対策機構が有効になるのかを確認するというものである．

調査の結果，オプションを指定してビルドしてもコンパイルエラーが発生するパッケージはなかった．また，SSP と Automatic Fortification はオプションを指定してビルドしても，生成されたバイナリでは対策機構が無効と判定された．そもそも，これらの 2 つの対策機構は dpkg コマンドでのビルド時にデフォルトで適用される設定になっているので，ソースコードが対策機構の適用条件を満たしていなかった可能性がある．Full RELRO と PIE は libc-bin と libc-dev-bin という 2 つのパッケージを除いて，生成されたバイナリで対策機構が有効になった．libc-bin と libc-dev-bin は，個別にビルド時に対策機構を適用しない設定にしていることが考えられる．

5. 考察

5.1 ソースコード解析とバイナリ解析の比較

本論文の調査では，各ディストリビューションに標準で含まれる ELF バイナリを解析した．コンパイル時に，ある対策機構 X を有効にするオプション指定をしても，その対策機構 X が適用されたバイナリが必ずしも生成されるわけではない．例えば，Ubuntu 14.04 32bit のソースパッケージ（cups-1.7.2）に含まれる cupsaccept は，ビルド時に Automatic Fortification を有効にするオプションが指定されるが，生成されたバイナリを解析すると関数の置換が行われていなかった．

5.2 PIE の適用率が低い理由

図 3 が示すように，ディストリビューションの種類によらず，PIE の適用率は低いことがわかった．この図は 64bit での結果を示しているが，4.1 で述べているように，32bit でも同様の結果であった．これは，32bit の Linux 環境における PIE はセキュリティの効果が高くないうえに，パフォーマンスへの悪影響があることが原因であると推察される．

セキュリティの効果の観点では，32bit における ASLR のエントロピーは低く，ブルートフォース攻撃によって回避されることが知られている [14]．

さらに，先行研究 [15] は，x86 対象の Ubuntu11.04 を実験環境として，SPEC CPU 2006 が提供する 19 個のバイナリについて PIE 適用時と非適時の実行時間の比較を行っており，適用時は非適用時と比較して平均約 10%のオーバーヘッドが生じたと述べている．

5.3 32bit 版と 64bit 版の比較

表 2 が示すように，32/64bit でバイナリ数に大きな差はなかった．また，対策機構の適用率にも大きな差はなかった．この結果から，今回調査したディストリビューションでは，32/64bit 版で同一のコンパイルオプションを適用していると推察できる．5.2 節で述べているように，32bit では PIE はセキュリティの効果が高くないうえに，パフォーマンスへの悪影響を招くが，64bit ではその限りではない．そのため，本来は 64bit 環境では PIE を有効にするべきである．

6. まとめ

本論文では，主要な 32/64bit の Linux ディストリビューションにおいて，コンパイラによるセキュリティ対策機構の適用状況を調査およびディストリビューション間で共通するバイナリを対象に対策機構の適用状況を比較した．

調査対象とした CentOS，Ubuntu，および，openSUSE のバイナリ数は，それぞれ，10,542 個，8,283 個，および，15,424 個で，総数 34,249 個である．

その結果，以下のことがわかった．

(1) ディストリビューションごとに，セキュリティ対策機構の適用状況が違っていた
(2) セキュリティ対策機構が，バイナリに一旦適用されたのち，後の世代で意図的に，非適用・弱化されたケースが散見された
(3) コンパイルオプション指定はされていたが，実際には，機能していないケースがあった

特に，3 については，対策機構が特定の条件下でしか適用されないことに起因するが，その適用条件については，プログラマへの周知が必要であると言える．

本論文の調査結果から，Ubuntu17.04 は，調査対象とした対策機構の適用率が 90%以上だったことから，セキュリティが高いと言える．

参考文献

[1] CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer, http://cwe.mitre.org/data/definitions/119.html
[2] A. Bittau, A. Belay, A. Mashtizadeh, D. Mazieres, and D. Boneh. Hacking blind. In Proc. of IEEE Symposium on Security and Privacy. 2014.
[3] K. Z. Snow, F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen, and A.R. Sadeghi. Just-in-time Code-Reuse: On the effectiveness of fine-grained address space layout randomization. In Proc. of IEEE Symposium on Security and Privacy. 2013.
[4] 近藤 秀太，渡辺 亮平，菅原 捷汰，横山 雅展，中村 慈愛，須崎 有康，齋藤 孝道，32/64 ビット Linux ディストリビューションバイナリの生成時期およびコンパイラセキュリティオプションの調査，暗号と情報セキュリティシンポジウム (2018)
[5] Balakrishnan G., Reps T., Melski D., and Teitelbaum T. WYSINWYX: What You See Is Not What You eXecute. In: Meyer B., Woodcock J. (eds) Verified Software: Theories, Tools, Experiments. Lecture Notes in Computer Science, vol 4171. Springer, Berlin, Heidelberg (2008)
[6] IPA オープンソース・ソフトウェアのセキュリティ確保に関する調査報告書, https://www.ipa.go.jp/files/000013695.pdf
[7] IPA ISEC セキュア・プログラミング講座:C/C++言語編 第 10 章 著名な脆弱性対策，https://www.ipa.go.jp/security/awareness/vendor/programmingv2/contents/c905.html
[8] [PATCH] Object size checking to prevent (some) buffer overflows, http://gcc.gnu.org/ml/gcc-patches/2004-09/msg02055.html
[9] Rober C.Seacourd，C/C++セキュアコーディング 第 2 版
[10] TRAPKIT checksec.sh, http://www.trapkit.de/tools/checksec.html
[11] Ubuntu Wiki: Security/Features, https://wiki.ubuntu.com/Security/Features
[12] enables PIE, which often doesn't work on odd platforms, https://bugs.freedesktop.org/show_bug.cgi?id=16621
[13] D-Bus 1.11.14 (2017-06-29), https://dbus.freedesktop.org/doc/NEWS

[14] Shacham, Hovav, et al. On the effectiveness of address-space randomization. In Proc. of the 11th ACM conference on Computer and communications security. ACM, 2004.
[15] Mathias Payer. Too much PIE is bad for performance. In Proc. of ETH Zurich, Department of Computer Science Technical Report 766. 2012