2W-01

Linuxディストリビューションにおける メモリ破壊攻撃への対策技術の適用状況の調査

王 氷‡ 角田 佳史‡ 堀 洋輔‡ 馬場 隆彰‡ 宮崎 博行‡ 近藤 秀太† 渡辺 亮平† 齋藤 孝道†

明治大学† 明治大学大学院‡

1 はじめに

ソフトウエアの脆弱性の一つに，メモリ破壊の脆弱性[1]が存在し，数多く報告されている．その一方で，メモリ破壊の脆弱性への対策技術も複数考案されてきた．しかし，いかに有効な対策技術が存在していても，対策技術がソフトウエアに適用されない限り，人々はその恩恵を受けることができない．そこで本論文では，先行研究[2]における対策技術の適用状況の報告では調査対象ではなかった FORTIFY_SOURCE を調査対象に加え，また，先行研究[2]よりも多くのソフトウエアを対象に，既存の Linux ディストリビューション内のソフトウエアに対するメモリ破壊の脆弱性への対策技術の適用状況を調査する．

2 メモリ破壊の脆弱性

多種多様なソフトウエアの脆弱性を識別している共通脆弱性タイプ一覧 CWE (Common Weakness Enumeration) 内に，メモリ破壊の脆弱性 (CWE-119) がある．これは，ソフトウエアが意図するバッファの境界外への読み書きを許してしまう脆弱性である．メモリ破壊の脆弱性を悪用すると，任意のコード実行，機密情報の読み取り，制御フローの変更およびシステムの破壊が可能になる．様々な対策技術がある一方で，CWE や脆弱性データベースの一つである NVD[3] (National Vulnerability Database) などにおいて，メモリ破壊の脆弱性は現在でも絶えず報告されている (図 1 参照)．

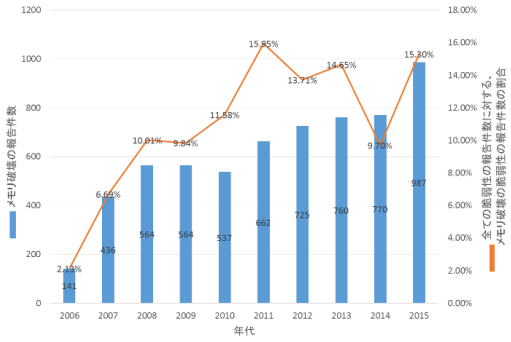

図 1. メモリ破壊の報告件数

Diffusion of Mitigation against Memory Corruption Attacks in Linux Distributions
‡Wang Bing ‡Sumida Yoshifumi ‡Hori Yosuke ‡Baba Takaaki
‡Miyazaki Hiroyuki †Kondo Shuta †Watanabe Ryohei
†Saito Takamichi
†Meiji University ‡Graduate School of Meiji University

3 関連知識

FORTIFY_SOURCE とは，バッファサイズを確認せずに文字列やメモリ操作を行う strcpy()や memcpy()などの脆弱性を招く関数群を，メモリ破壊の脆弱性の一つであるバッファオーバーフローを簡易に検査する代替関数（以下，検査用関数という）に置き換える gcc の機能である[4][5]. gcc 4.0 以降，glibc 2.3.4 以降で利用できる．

gcc でのコンパイル時に_FORTIFY_SOURCE が指定される場合，gcc は，コピー先のバッファとコピー元のデータの状態に応じて，以下のいずれかの挙動をする．

1) コピー先のバッファのサイズとコピー元のデータのサイズを判定でき，コピー先のバッファがコピー元のサイズより長い場合，検査用関数への置き換えは行わない．
2) コピー先のバッファのサイズを判定でき，コピー元のデータのサイズを判定できない場合，検査用関数への置き換えを行う．
3) コピー先のバッファのサイズとコピー元のデータのサイズを判定でき，コピー先のバッファがコピー元のサイズより短い場合，検査用関数への置き換えを行うとともに警告を表示する．
4) コピー先のバッファのサイズとコピー元のデータのサイズの両方を判定できない場合，検査用関数への置き換えは行わない．

実行時に検査用関数がバッファオーバーフローを検知した場合，プロセスは異常終了する．

4 対策技術の適用状況

本論文では，Ubuntu 15.10 64bit (ubuntu-15.10-desktop-amd64.iso)，Debian 8.2.0 64bit (debian-8.2.0-amd64-DVD-1.iso)，および CentOS 7 64bit (CentOS-7-x86_64-DVD-1503-01.iso) 内の ELF 形式の実行ファイル (以下，実行ファイルという) を対象に，SSP，NX (execstack)，PIE，RELRO，および FORTIFY_SOURCE の適用状況を調査した．各ディストリビューションのインストール時のパッケージ選択はデフォルトのままとした．Ubuntu と Debian に関しては，apt-get を用いて build-essential をインストールし，tasksel を用いてデスクトップカスタマイズ用のパッケージ以外のパッケージを全てインストールした．

Copyright ©2016 Information Processing Society of Japan.
All Rights Reserved.

また CentOS に関しては，yum の groupinstall でインストール可能な全てのグループ内のパッケージをインストールした．対策技術の適用の有無は，実行ファイルの ELF ヘッダやシンボル情報から判定を行った．特に，SSP と FORTIFY_SOURCE に関しては，シンボル情報がある実行ファイルのみを対象とした．

さらに，文献[6]で示されているバッファオーバーフロー脆弱性のある関数（以下，脆弱関数という）が実行ファイル内で呼び出されているかに関しても，実行ファイルを元に調査した．

表 1～5 に，対策技術の適用の有無と，脆弱関数の呼び出しの有無で，実行ファイルを分類した結果を示す．ただし，表 3 は動的共有オブジェクトを調査の対象から外した結果である．また，表 4 の「適用」は Full RELRO が適用されていることを意味し，「非適用」は Partial RELRO が適用されているもしくは RELRO が適用されていないことを意味する．

表 1. SSP の適用状況

	脆弱関数の呼び出し	対策技術		対策技術の適用率
		適用	非適用	
Ubuntu	なし	3047 個	2356 個	75.31%
	あり	5108 個	318 個	
Debian	なし	1881 個	928 個	76.81%
	あり	2702 個	456 個	
CentOS	なし	1748 個	698 個	85.37%
	あり	3645 個	226 個	

表 2. NX の適用状況

	脆弱関数の呼び出し	対策技術		対策技術の適用率
		適用	非適用	
Ubuntu	なし	5445 個	2 個	99.94%
	あり	5422 個	4 個	
Debian	なし	2850 個	1 個	99.97%
	あり	3157 個	1 個	
CentOS	なし	2459 個	2 個	99.97%
	あり	3871 個	0 個	

表 3. PIE の適用状況

	脆弱関数の呼び出し	対策技術		対策技術の適用率
		適用	非適用	
Ubuntu	なし	229 個	848 個	21.10%
	あり	438 個	1646 個	
Debian	なし	98 個	692 個	19.82%
	あり	318 個	991 個	
CentOS	なし	141 個	483 個	31.63%
	あり	674 個	1279 個	

表 4. RELRO の適用状況

	脆弱関数の呼び出し	対策技術		対策技術の適用率
		適用	非適用	
Ubuntu	なし	627 個	4820 個	14.97%
	あり	1001 個	4425 個	
Debian	なし	325 個	2526 個	15.39%
	あり	600 個	2558 個	
CentOS	なし	271 個	2190 個	16.74%
	あり	789 個	3082 個	

表 5. FORTIFY_SOURCE の適用状況

	脆弱関数の呼び出し	対策技術		対策技術の適用率
		適用	非適用	
Ubuntu	なし	922 個	4481 個	41.74%
	あり	3598 個	1828 個	
Debian	なし	521 個	2288 個	42.67%
	あり	2025 個	1133 個	
CentOS	なし	691 個	1755 個	59.05%
	あり	3039 個	832 個	

表 1～5 より，各ディストリビューション内の実行ファイルに対する対策技術の適用に関して，以下のことがわかる．SSP と NX に関しては，各ディストリビューションにおいて，対策技術が適用されている実行ファイルの方が多い．PIE と RELRO に関しては，各ディストリビューションにおいて，対策技術が適用されている実行ファイルの方が少ない．FORTIFY_SOURCE に関しては，Ubuntu と Debian においては対策技術が適用されている実行ファイルの方が少なく，CentOS においては対策技術が適用されている実行ファイルの方が多い．

5 まとめ

本論文では，比較的新しい Linux ディストリビューション内の実行ファイルに対して，メモリ破壊攻撃への対策技術がどの程度適用されているかを示した．調査結果から，比較的新しい Linux ディストリビューションにおいて，既存の対策技術が非適用かつバッファオーバーフロー脆弱性がある関数を呼び出す可能性のある実行ファイルが一定数存在することがわかった．

6 参考文献

[1] CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer https://cwe.mitre.org/data/definitions/119.html
[2] National Vulnerability Database https://web.nvd.nist.gov/view/vuln/statistics
[3] Takamichi Saito, Hiroyuki Miyazaki, Takaaki Baba, Yoshifumi Sumida & Yosuke Hori (2015). Study on diffusion of protection/mitigation against memory corruption attack in linux distributions, Proc. of the 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS) 2015
[4] FEATURE_TEST_MACROS(7) http://man7.org/linux/man-pages/man7/feature_test_macros.7.html
[5] Robert C. Seacord (2014) 『C/C++セキュアコーディング 第 2 版』(歌代和正・久保正樹・椎木孝斉訳) JPCERT コーディネーションセンター
[6] John Viega & Gary McGraw (2006) 『Building Secure Software―ソフトウェアセキュリティについて開発者が知っているべきこと』(齋藤孝道・河村政雄・武舎広幸訳) オーム社

Copyright ©2016 Information Processing Society of Japan.
All Rights Reserved.