コンパイラの最適化が引き起こす脆弱性ガード消失のバイナリ比較による検出

Detection of Vulnarability Guard Elimination by Compiler Optimization Based on Binary Comparison

東 優花　　石浦 菜岐佐
Yuka Azuma　　Nagisa Ishiura

関西学院大学 理工学部　School of Science and Technology, Kwansei Gakuin University

1　はじめに

プログラムの脆弱性は社会的にも深刻な問題を引き起こしており，これを防ぐために徹底的なコード監査や形式的手法による検証等が行われる．しかし，ソースコードレベルで対策を行ったとしても，コンパイラの最適化が意図しない脆弱性を引き起こすことが指摘されている [1]．本稿ではこのような脆弱性のうちガードの消失に着目し，これをバイナリコードの比較により検出する手法を提案する．

2　コンパイラの最適化によるガード消失

本稿ではプログラム中の脆弱性を防止するためのコードを「ガード」と呼ぶ．コンパイラの最適化によりガードが消失するプログラムの例を図 1 に示す．(a) の 4 行目は，size がオーバーフローした場合のメモリ割り当て防止を意図したガードであるが，C 言語では符号付きオーバーフローは未定義動作であるため，コンパイラの最適化がこのガードを削除することがある．(b) では 9 行目の NULL ポインタ逆参照を 8 行目で防止しようとしているが，7 行目で p を使用していると，最適化が 8 行目のコードを削除することがある．

```
SOF.c
1: int sub (int n){
2: ...
3:   int size = BASE + n;
4:   if(size < n) error();
5:   char* p = malloc(size);
6:   ...
7: }
```

(a) SOF: 符号付きオーバーフロー

```
NPD.c
1: typedef struct{
2:   int a[SASIZE];
3:   int x;
4: } str_t;
5: ...
6: int sub(str_t p){
7:   int y = p -> x;
8:   if(p==NULL) error();
9:   int z = p -> a;
10: }
11: ...
```

(b) NPD: NULL ポインタ逆参照

図 1　コンパイラの最適化が削除するガードの例

3　バイナリ比較に基づくガード消失の検出

本稿では，コンパイラの最適化によるガードの消失を，バイナリ比較によって検出する手法を提案する．本手法では図 2 に示すように，与えられたプログラムに対して通常通り最適化を行った場合と，ガードを削除する可能性のある最適化を抑止した場合に生成されるコードを比較することにより，ガード消失の検出を試みる．

これらの最適化抑止はコードを大きく変化させることがあるため，単純なコード比較ではガード消失のみを検出することができない．そこで本手法では，ガードのエラー処理は通常サブルーチン呼び出しで行われることに着目し，コード中の call 命令の比較に基づいて検出を行う．図 2 のように，各関数内の call 命令の対応を取り，対応が取れなかったものを抽出する．この call 命令がエラー処理を呼び出している場合には，ガード消失の可能性があるものと判定する．

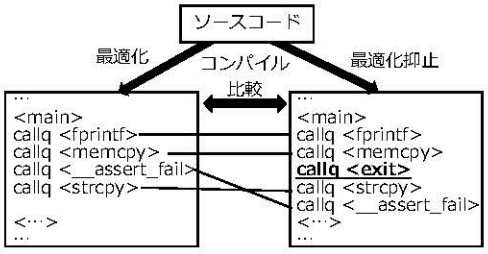

図 2　バイナリコードの比較法

4　実験結果

本手法に基づくテストシステムを Perl 5 で実装し，テストを行った．結果を表 1 に示す．対象プログラムは coreutils-8.24 の cat, ls, echo, およびコンパイラ mcc であり，“KLOC” はソースコードの行数 (×千行) である．デフォルトのコンパイルは GCC-7.3.0 の -O2 オプションで行い，“SOF”, “NPD” はそれぞれ符号付きオーバーフロー，NULL ポインタ逆参照に関する最適化を抑止するオプション-fwrapv, -fno-delete-null-pointer-checks を用いた結果である．“差分” は本手法による不一致の検出数，“検出” は目視によって確認したガードの消失の件数である．今回の実験では誤検知はなく，1 件のガード消失を検出できた．GCC は符号付きオーバーフローに関するコード削除を警告するオプション -Wstrict-overflow を実装しているが，上記のガード消失は警告されなかった．

表 1　実験結果

プログラム	KLOC	SOF		NPD	
		差分	検出	差分	検出
cat	0.2	0	0	0	0
ls	0.2	0	0	0	0
echo	4.9	0	0	0	0
mcc	1.2	1	1	0	0

5　むすび

本稿では，コンパイラの最適化による脆弱性ガードの削除を検出する手法を提案した．消失した call 命令がガードに属するかの判定の自動化と，実際的なプログラムでの実験が今後の課題である．

参考文献

[1] Vijay D'Silva, et al.: “The Correctness-Security Gap in Compiler Optimization,” in *Proc. IEEE Security and Privacy Workshops*, pp. 73–87 (May 2015).