視覚化に基づくプログラムの脆弱性学習支援システムの構築

Construction of Learning Support System for Software Vulnerability Based on Program Visualization

瀧 嘉人[*1], 野口 靖浩[*2], 小暮 悟[*2], 山下 浩一[*3], 小西 達裕[*2], 伊東 幸宏[*4]
Yoshito TAKI[*1], Yasuhiro NOGUCHI[*2], Satoru KOGURE[*2],
Koichi YAMASHITA[*3], Tatsuhiro KONISHI[*2], Yukihiro ITOH[*4]
[*1] 静岡大学大学院 総合科学技術研究科
[*1]Graduate School of Integrated Science and Technology, Shizuoka University
[*2] 静岡大学 情報学部
[*2]Faculty of Informatics, Shizuoka University
[*3] 常葉大学 経営学部
[*3]Faculty of Business Administration, Tokoha University
[*4] 静岡大学
[*4]Shizuoka University
Email: [*1]taki.yoshito.14@shizuoka.ac.jp, [*2]{konishi, kogure, noguchi}@inf.shizuoka.ac.jp
[*3]yamasita@hm.tokoha-u.ac.jp, [*4]itoh@inf.shizuoka.ac.jp

あらまし：現在，ソフトウェアの脆弱性に対する攻撃の増加が問題となっている．C 言語は不用意に用いると脆弱性を生みやすいプログラミング言語であるが，脆弱性についての理解を深めることは，多様な知識が必要となるため多くの学習者にとって容易ではない．そこで本論文では，脆弱性がプログラムに異常な振る舞いを引き起こす仕組みを可視化することによって脆弱性の学習を支援するシステムを構築する．
キーワード：プログラミング学習支援，脆弱性学習支援，プログラム視覚化

1. はじめに

近年，ネットワーク化されたソフトウェアシステムへの依存度が高まるにつれ，そのようなシステムを狙った攻撃の数は増え，ソフトウェアの脆弱性に関する報告は驚くべき速さで増えている(1)．脆弱性とは，プログラムや設定上の問題に起因するセキュリティ上の「弱点」であり，脆弱性が残された状態でコンピュータを利用していると，侵入されたり，不正アクセスに利用されたり，ウイルスに感染したりする危険性がある(2)．しかしながら，国内では上記のような脆弱性に対応する技術者は不足しており，多くの企業がセキュリティ人材の不足感を感じていることが報告されている(3)．演習課題によく採用される C 言語は不用意に用いると脆弱性を生みやすいプログラミング言語(4)であり，その学習には OS，機械語，ネットワークなど多様な知識が必要なためその学習は多くの学習者にとって簡単なものではない．

そこで本論文では，脆弱性がプログラムに異常な振る舞いを引き起こす仕組みを可視化することによって脆弱性の学習を支援するシステムを構築する．今回は数ある脆弱性の内バッファオーバーフローの学習を支援するシステムを構築する．

2. 関連研究・先行研究

2.1 関連研究

プログラムの脆弱性を体系的に学習する Web コンテンツとして AppGoat(5)がある．AppGoat ではアプリケーションの異常な振る舞いが起きた瞬間を観測させ脆弱性の影響の理解を促すが，ソースコードと結び付けて異常な振る舞いが起きる過程を観測することは出来ない．またプログラムの挙動を可視化するシステムの主な例として PROVIT(6)がある．PROVIT ではプログラムの振る舞いをソースコードに結び付けてステップごとに可視化することが出来るが，異常な振る舞いの観測や，異常な振る舞いが起きる仕組みの観測はサポートしていない．

2.2 先行研究

著者らが開発した TEDViT(7)は，2.1 節で述べた PROVIT と同様にプログラムの振る舞いをソースコードに結び付けてステップごとに可視化することが出来る．さらに TEDViT は上記の機能に加え以下に示す 2 つの機能を持つ．

(1) メモリ領域をステップごとに可視化する機能
(2) ルールを基に教師の説明意図を反映し可視化する機能

(2)における説明意図とは対象のプログラムの理解を助けるための視覚化の方法であり，教師が課題ごとにルールを作成する．現状の TEDViT も PROVIT と同様に，異常な振る舞いが起こる仕組みの観測をサポートしているわけではない．しかし TEDViT では上記に述べた特有の機能を用いることにより，異常な振る舞いが起こる仕組みの可視化が行える．

3. 基礎的考察

どのようなアプローチで脆弱性の学習を支援するのか C 言語プログラミングの講義を行う教師と議論を交わした．今回はバッファオーバーフローに関係する脆弱性に関して考察を行った．

3.1 脆弱性に関して

バッファオーバーフローとはプログラムが管理し

ているメモリ上に確保するバッファ領域に対して，脆弱性のある実装をしているとバッファ領域の上限を超えた部分に情報を格納してしまうことである．これによりプログラムが意図した通りに動作しなくなってしまう場合がある．またバッファオーバーフローを利用しプログラムのリターンアドレスなどを書き換えることにより任意のコードを実行させることなどが出来る．これらの動作を理解するには，脆弱性がプログラムにどのような振る舞いを引き起こすのか観測する必要がある．特に脆弱性が起きた際のメモリ領域等を観測する必要がある．

3.2 学習が困難な点

3.1 節で述べた事柄は，GDB(Gnu DeBugger)を用いることで観測できる．しかし GDB を用いて得られる情報を組み合わせて脆弱性の理解に必要な情報として確認することは，多くの学習者にとって簡単なことではない．情報系大学のカリキュラム構成から以下の点で確認が困難であると考察した．

(A) プログラムが利用するメモリ領域の構造
(B) レジスタ情報
(C) プログラムによるメモリ領域内の変化
(D) プログラムの変数とメモリ領域の関係
(E) メモリ領域特有の表示形式

4. システムの構築

今回はバッファオーバーフローに関する脆弱性を対象として，脆弱性がプログラムの異常な振る舞いを引き起こす仕組みを可視化することにより脆弱性の学習を支援するシステムの構築を行う．システムは先行研究である TEDViT を改良して構築する．具体的には，バッファオーバーフローに関係する脆弱性を理解するのに必要な GDB を用いて得られる情報を自動的に表示できるように TEDViT を拡張した．さらにそれらの表示される情報に対して 3.2 節で述べた(A)から(E)の学習が困難な点を支援できるように，TEDViT のルールの拡張を行った．

実際にバッファオーバーフローを引き起こす可能性のある関数(strcpy)を利用して，バッファオーバーフローを引き起こし，リターンアドレスを書き換えた際のシステムの挙動を示す．システムはプログラムの挙動とその際のメモリ領域をステップごとに可視化する．システムが表示するプログラムの挙動を図 1 に，メモリ領域を図 2 に示す．図 1，2 に示すようにシステムは脆弱性の理解に必要な情報を可視化し，その情報に対して注目箇所へのメッセージの提示やメモリ上の値の変化の強調表示など，教師の意図を反映し学習を支援する可視化を行う．

5. むすび

今回バッファオーバーフローに関する脆弱性を対象に，脆弱性がプログラムに異常な振る舞いを引き起こす仕組みを可視化することによって学習を支援するシステムの構築を行った．今後，システムの有効性や課題を確認するため，静岡大学で開催予定の Basic SecCap/サイバー攻防基礎演習にて評価を行う．

図 1　プログラムの挙動

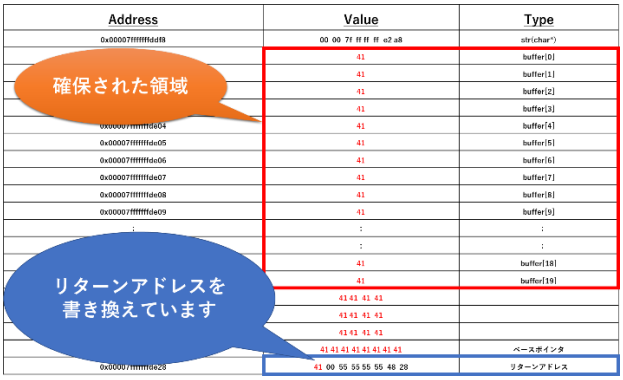

図 2　メモリ領域

参考文献

(1) JPCERT コーディネーションセンター：“CERT C コーディングスタンダード”，https://www.jpcert.or.jp/sc-rules/ (参照 2019.6.13)
(2) IPA 独立行政法人 情報処理推進機構技術本部 セキュリティセンター：“セキュア・プログラミング講座 第 1.1 版”，https://www.ipa.go.jp/files/000059838.pdf (参照 2019.6.13)
(3) IPA 独立行政法人 情報処理推進機構技術本部 セキュリティセンター:“「情報セキュリティ人材の育成に関する基礎調査」報告書について”，https://www.ipa.go.jp/security/fy23/reports/jinzai/ (参照 2019.6.8)
(4) IPA 独立行政法人 情報処理推進機構技術本部 セキュリティセンター：“セキュア・プログラミング講座 C/C++言語編”, https://www.ipa.go.jp/security/awareness/vendor/programmingv2/cc01.html (参照 2019.6.13)
(5) IPA 独立行政法人 情報処理推進機構技術本部 セキュリティセンター：“脆弱性体験学習ツール AppGoat(個人学習向け)：ツール概要”，https://www.ipa.go.jp/security/vuln/appgoat/ (参照 2019.6.13)
(6) 松村 和哉, 渡部 治朗, 寺内 俊, HE Aiguo：“PROVIT：ソフトウェア可視化手法を用いた初心者向け C 言語教育ツール”，電子情報通信学会技術研究報告, vol.109，no.268，pp.41-46 (2009)
(7) Satoru KOGURE, Ryota FUJIOKA, Yasuhiro NOGUCHI, Koichi YAMASHITA, Tatsuhiro KONISHI, Yukihiro ITOH, “Code Reading Environment by Visualizing both Variable’s Memory Image and Target World’s Status”, Proceedings of ICCE2014, pp.343-348 (2014)