報道各位

2020年2月21日
株式会社インプレスR&D
https://nextpublishing.jp/

不正なコードからコンピュータを守るサイバーセキュリティ技術

『はじめて学ぶバイナリ解析』発行

セキュリティの基礎知識を身につける入門書！

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『はじめて学ぶバイナリ解析』(著者:小林佐保・岡田怜士・浅部佑・満永拓邦)を発行いたしました。

『はじめて学ぶバイナリ解析』
https://nextpublishing.jp/isbn/9784844378495

著者:小林佐保・岡田怜士・浅部佑・満永拓邦
小売希望価格:電子書籍版　1600円(税別)／印刷書籍版　2000円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／モノクロ／本文178ページ
ISBN:978-4-8443-7849-5
発行:インプレスR&D

＜＜発行主旨・内容紹介＞＞

近年、総務省や経済産業省では、サイバーセキュリティ人材育成を社会的な課題とする重要施策の一環として取り扱っています。しかし、「セキュリティの基本的な解説書」が、英語・国語のような基礎的な科目と比較して、現状十分に揃っていません。そこで、まずはセキュリティの基礎技術であるバイナリの入門的な内容をまとめました。

バイナリ解析が直感的に理解しにくいことを考慮し、プログラミングに詳しくない大学1・2年生や新社会人でも理解できるよう、各章での説明はできるだけやさしい表現を用いました。他方、深い技術的な内容はそれほど取り上げていません。

また、基礎理論の習得だけではなく、自ら演習をすることで、より理解を深める点を重視しています。演習の題材

では、バッファオーバーフローと呼ばれる脆弱性（セキュリティホール）を取り上げます。その問題を解く演習により、本書の構成を実行ファイルの解析であるバイナリ解析技術の基礎を得られるようにしています。

本書の目的は、バイナリ解析の基本を理解していただくことです。高度な専門書を読むための前提知識を取得する入門者向けであり、コンピュータが動作する基盤となる原理や仕組みに読者の方々が興味を持つ端緒になれば幸いです。

（本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。）

ビットとバイトと16進数

3章　ASCIIコードとバイトオーダ

3-1　ビットとバイトと16進数

前章ではバイナリエディタを使って16進数で表記されたバイナリコードを確認しました。2進数と16進数の対応を見てみましょう。

2進数というのは、0と1のみを用いて数字を表す方法です。0と1しか扱えないので、各桁で2に達すると位（くらい）が上がります。1桁目が1の位で、2桁目が2の位、3桁目が4の位…というようになります。2進数であることを明示したいとき、数字の末尾に「binary」の「b」を書くことがあります。私たちが普段用いている10進数と比較すると、表3-1のようになります。

表3-1　10進数と2進数

10進数	2進数
0	0b
1	1b
2	10b
3	11b
4	100b

16進数というのは、1の位の次の位が、16の位になっているような数字の書き方のことをいいます。10進数では、0から9までの10通りの数字を用いて、1つの位を表現していますが、16進数では0から15までの16通りの数字を1つの位で表現しますので、0から9まででは足りません。そこで、AからFまでのアルファベット6文字を、10から15までの数字を表す記号として用います。16進数の表記であるということを明確にするために、16進数の場合は頭に0xをつけることがあります（表3-2）。

2進数と16進数を並べてみましょう（表3-3）。

このように対応付けると、2進数の4桁が表す0から15までの数字が、ちょうど16進数の1桁に対応していることになります。たとえば、日常的に利用している10進数の10は、表のグレーの部分で、2進数では1010b、16進数では0xAで表現されます。

2進数の1桁を1ビット（bit）、1ビットが8つ集まったものを1バイト（byte）と呼び、32ビットのコンピュータであれば4バイトをひとまとまり、64ビットであれば8バイトをひとまとまりとして情報を処理します。1バイトは2進数であれば8桁、16進数であれば2桁なので、32ビットのコンピュータは16進数の8桁をひとまとまりとして処理します（表3-4）。

表3-2　10進数と16進数

10進数	16進数
0	0x0
1	0x1
2	0x2
…	…
9	0x9
10	0xA
11	0xB
…	…
15	0xF
16	0x10
17	0x11
…	…

表3-3　10進数と2進数と16進数

10進数	2進数	16進数
0	0b	0x0
1	1b	0x1
2	10b	0x2
…	…	…
9	1001b	0x9
10	1010b	0xA
11	1011b	0xB
…	…	…
15	1111b	0xF
16	10000b	0x10
17	10001b	0x11
…	…	…

表3-4　2進数、16進数、バイトとビットの関係

2進数	0111	1010	0000	1111	0100	1001	1011	0000
16進数	0x7	0xA	0x0	0xF	0x4	0x9	0xB	0x0
1バイト	0x7A		0x0F		0x49		0xB0	
32ビット	0x7A0F49B0							

スタック

します。

スタックデータ構造にデータを追加することをpush（プッシュ）、削除することをpop（ポップ）といいます。

4-2　スタック

たとえば、次のような順にpushとpopの操作をすることを考えてみましょう。

```
push 1
push 2
push 3
push 4
pop
pop
push 5
```

この操作が終わったとき、スタックはどのような状態になっているでしょうか？

最初、スタックは空の状態から始まります（図4-2）。

図4-2　スタック図1

push 1をすると、上に1が積み重なります（図4-3）。

図4-3　スタック図2

push 2、push 3、push 4を同様に行うと、図4-4のような状態になります。

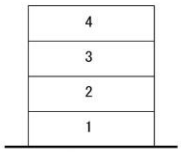

図4-4　スタック図3

この後popをすると、最後に追加され一番上にある4が取り出されます（図4-5）。
もう一度popをすると、今ある中で一番上にある3が取り出されます（図4-6）。
この状態でpush 5をすると、一番上に5が追加されます（図4-7）。
よって、図4-7のような状態が、問題の答えになります。

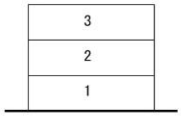

図4-5　スタック図4

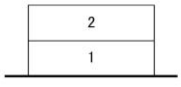

図4-6　スタック図5

演習：レジスタの動きを見る

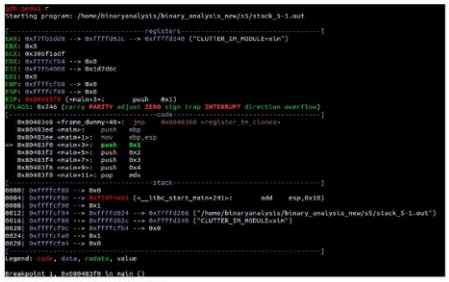

図5-7　runコマンド入力後のgdb-pedaの画面

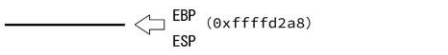

図5-8　スタック図1

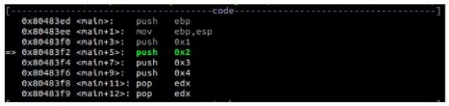

図5-9　gdb-pedaのcodeセクション

実行前と比べて、codeセクションが1行分ずれて、EIPの指すアドレスが1行移動したことがわかるでしょうか。レジスタセクションのEIPの表示の方は、

```
EIP:0x80483f2
```

となっていて、やはり一致していることがわかります。

EBPとESPは、

```
EBP:0xffffcf88
ESP:0xffffcf84
```

となっています。図にすると図5-10のような状態です。

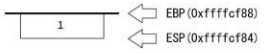

図5-10　スタック図2

先ほどと比べると、EBPは変わっていませんが、ESPが4小さくなっています。このことから、4バイト分pushされたということがわかります。push 0x1が実行されたことを考えると、スタックには0x1が入っていてほしいところですが、gdb-pedaのスタックのセクションを見てみると、確かに0x1の文字が見えます。

この後の3つのpushの命令も同様に確認してみましょう。ステップ実行をするたびに、EIPとESPが更新されるのがわかるかと思います。

popの命令では、pushと逆の操作が行われるので、ESPの値が4バイトずつ増え、EBPに近づいていくのが確認できるはずです。EIPはpush命令かpop命令かによらずに、少しずつ増えていきます。

終了したい場合はqコマンドで終了できます。

レジスタの感覚がつかめてきたでしょうか。興味がある方は、今まで扱ってきたバイナリをgdb-pedaで実行してレジスタの動きを確認してみてください。

5-5　条件分岐

C言語のif文のような条件分岐は、アセンブリ言語ではジャンプ命令と呼ばれる命令群で実装されています。ジャンプ命令自体にはif文のように条件をチェックする機能はなく、チェックできるのはフラグレジスタの状態だけです。事前に、分岐の条件に応じてフラグレジスタを更新しておいてから、ジャンプ命令でフラグレジスタを参照して分岐します。

フラグレジスタの更新にはcmpなどの命令が、ジャンプ命令にはjeなどの命令が用いられます。たとえばje命令は、図5-11のようにゼロフラグ（ZF）が0か1かで分岐処理が行われて、0のときにジャンプが実行されます。

演習：画面に“Hi”と表示しよう

```
global main

main:
```

バッファはスタックに確保しましょう。つまり、スタックに"Hi"をpushし、スタックへの第2引数としてポインタを指定します。その際、"Hi"はASCIIコードで指定する必要があります（表6-6）。

表6-6　表示したい文字とASCIIコード

表示する文字	ASCIIコード
H	0x48
i	0x69

例によって、数字で指定したものを文字列として出力する型の読み換えがあるので、スタックにpushするものはリトルエンディアンの順番に並び変える必要があります。そのため、バッファにpushするものは0x6948になります。ただし、pushするものは4バイト単位なので、0x6948と書くと0x00006948と自動的に読み換えられます。ここでは混乱を避けるため、最初から0x00006948と書いてしまいましょう。

```
push 0x00006948
```

システムコールのwriteを指定するために、EAXをwriteのシステムコール番号の4にセットしましょう。

```
mov eax, 0x4
```

同様に第1引数と第3引数も指定しましょう。今回の出力先は標準出力の0x1にします。

```
mov ebx, 0x1
```

第2引数を指定するために、今のスタックの状況を考えましょう（図6-7）。

このスタックの状況は、メモリの様子で見ると0x00006948はリトルエンディアンで格納されているので、図6-8のようになっているはずです。

ポインタを与えたときには、指定した位置からアドレスの大きい方の向きに読んでいくので、このHiの領域を出力するには、Hiの書かれた領域の先頭であるESPの位置を指定すればいいでしょう。よって、ECXはこうなります。

図6-7　スタック図

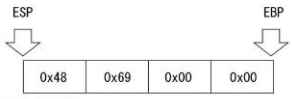

図6-8　メモリの状態

```
mov ecx, esp
```

EDXは出力する文字数です。2文字でも3文字でも4文字でも構いませんが、4バイト単位で扱うことが多いので今回は4バイトにしましょう。1桁なので、16進数に変換しても4は0x4です。

```
mov edx, 0x4
```

この状態でint 0x80を呼べば、writeが呼ばれてHiを出力してくれるはずです。まとめると、以下のようになります。

プログラム hi.s（~/binary_analysis/s6/）

```
global main

main:
  push 0x00006948
  mov eax, 0x4
  mov ebx, 0x1
  mov ecx, esp
  mov edx, 0x4
  int 0x80
```


＜＜目次＞＞

＜＜著者紹介＞＞

小林 佐保（こばやし さほ）
千葉大学大学院融合理工学府数学情報科学専攻。2016 年、ハワイ大学との交換留学協定プログラムにてオートマトンを研究。2017 年、千葉大学セキュリティバグハンティングコンテスト最優秀賞と千葉大学理学部後援会長賞を受賞。2018 年、トビタテ日本代表プログラムにてシンガポール国立大学で論理学を研究。2018 年、GoogleSWE インターンにて機械学習によるデータ解析業務を行った。

岡田 怜士（おかだ さとし）
東京大学工学部計数工学科所属。2016 年、東京大学理科 1 類に入学。東京大学情報学環セキュア寄付講座主催のサイバーセキュリティトレーニング参加をきっかけにセキュリティ技術に興味を持つ。その後、東京大学情報学環で受託している情報処理推進機構産業サイバーセキュリティセンターの人材育成事業の補助アルバイトとして従事し、セキュリティ技術を日々磨いている。

浅部 佑（あさべ ゆう）
東京大学工学部電子情報工学科所属。東京大学情報学環セキュア寄付講座主催のプログラムに参加後、セキュリティの奥深さに惹かれ、研究や人材育成事業に携わり始める。アプリ・Web サービス等の開発の経験を活かし、2017 年、株式会社 Gracia を共同創業し、取締役 CTO として EC サービスをフルスクラッチで作成する。現在は勉学と新しい技術の習得に幅広く励んでいる。

満永 拓邦（みつなが たくほう）

東京大学大学院情報学環特任准教授。京都大学情報学研究科修了後、民間企業のセキュリティソリューション事業部にて、ペネトレーションテストやセキュリティインシデント対応などの業務を行う。2011 年、JPCERT/CC 早期警戒グループに着任し、標的型攻撃などサイバー攻撃に関する分析等に従事する。2015 年、東京大学情報学環セキュア情報化社会研究寄付講座特任准教授として着任し、サイバー攻撃防御手法の研究やセキュリティ人材育成、ブロックチェーンなどの研究を行う。『サイバー攻撃からビジネスを守る』や『CSIRT』(ともに NTT 出版)等の書籍の共著・監修も行っている。

＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※ 各ストアでの販売は準備が整いしだい開始されます。

※ 全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/

株式会社インプレス R&D(本社:東京都千代田区、代表取締役社長:井芹昌信)は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム(またはメソッド)の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド(POD)による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス(本社:東京都千代田区、代表取締役:唐島夏生、証券コード:東証 1 部 9479)を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp