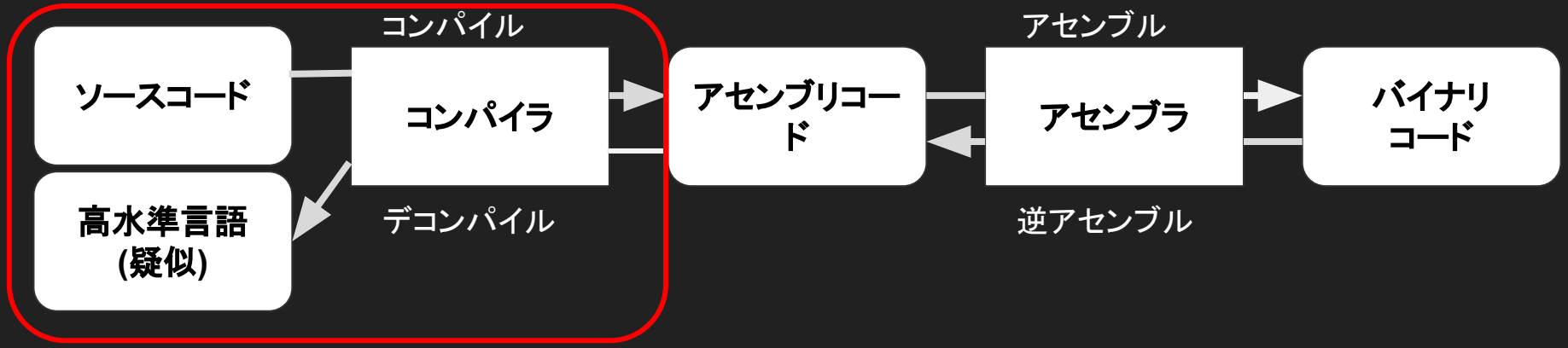
バイナリ後半

～ghidraでデコンパイルしてみる～

前回の振り返り

デコンパイルとディスアセンブルの流れ



今回はデコンパイルがどうなるかみる

リバースエンジニアリング

（前回ちゃんと説明してなかったので）

ハードウェアやソフトウェアを分析して、その構成や機能を明らかにする技術のこと。（今回はソフトウェアに関すること）

ソフトウェアのクラックなど違法な用途に用いられることもあるが、マルウェア解析や脆弱性の発見などの用途にも勿論用いられる。

ハードウェアだけど最近あった面白いやつ

スイスの大学生がiphoneXのLightningポートをUSB-Cポートに改造した

オークションで86万ドル日本円で約980万円で落札された

https://news.yahoo.co.jp/articles/f3c26bf6c56b3a783b356bebb071dd7ec0abf7fe

ghidraの説明

NSA(アメリカの国家安全保障局)がGhidraというリバースエンジニアリングツールを公開!

これまでIDA一強だったが無料で高機能なデコンパイラが使えるghidraが登場！そのほかの機能も充実している。

（多くのアーキテクチャや実行ファイル形式に対応。解析の自動化を行うスクリプト機能も充実している。一応デバッガもあるが...）

簡単なCプログラムを
デコンパイル＆ディスアセンブル
グラフビューも見る

記法について

記法には、Intel記法とAT＆T記法がある。

大きな違いとしてオペランドの順番が異なる

オペコード：命令が行う操作の種類を指定する部分

オペランド：オペコードによる操作の対象となる部分

Intel記法

mov eax, 5

AT＆T記法

mov $5, %eax

C言語

eax = 5;

アセンブリ命令

命令	種類	操作	ファーマット
MOV	データ転送命令	値をコピーする	MOV dest, src
PUSH	データ転送命令	スタックに値を保存する	PUSH value
POP	データ転送命令	スタックから値を取得する	POP dest
ADD	算術命令	加算	ADD dest, src
SUB	算術命令	減算	SUB dest, src
MUL	算術命令	符号なし乗算	MUL dest
DIV	算術命令	符号なし除算	DIV dest
XOR	論理演算命令	ビットごとの排他的論理和演算	XOR dest, src
CALL	制御転送命令	関数の呼び出し	CALL function
LEA		実行アドレスをロードする	LEA dest, src
RET		関数から呼び出し元に戻る	RET

src(source):出所
dest(destination):行先

アセンブリ命令

命令	種類	操作	フォーマット
SHL	論理演算命令	論理左シフト	SHL dest, src
CMP	算術命令	src1をsrc2と比較し、結果に従ってEFLAGSレジスタをセットする	CMP src1 src2
JNZ	分岐命令	ゼロでない場合(ZF = 0)はジャンプ	JNZ address
JL	分岐命令	より小さい場合(SF = 0)はジャンプ	JL address
JZ	分岐命令	ゼロの場合(ZF=1)はジャンプ	JZ address
JA	分岐命令	より上の場合(CF = 0 & ZF = 0)はジャンプ	JA address
JNE	分岐命令	等しくない(ZF = 0)	JNE address
JMP	分岐命令	無条件ジャンプ	JMP address
JGE	分岐命令	より大きいか等しい場合(SF = 0F)はジャンプ	JGE address

名称	概要
ZF	演算結果が0のときセットされる
SF	演算結果が負のときセットされる
CF	符号なし演算結果がオーバーフローしたときセットされる
0F	符号付き演算でオーバーフローしたときセットされる

説明（if、switch）

MOV dword ptr [EBP + local_8], EAXでatoi関数で数値に変換した入力値をlocal_8に格納

CMP dword ptr [EBP + local_8],0x0で比較。

JNZ　0であれば次の処理実行puts関数を実行0で無ければジャンプ

次のCMPで0x1と比較

JL　小さければジャンプ

（計算結果が負であればジャンプCMPは内部的にSUBと同じ）

CMPと条件フラグとジャンプ

switchはcaseの数が増えるとジャンプテーブルというものを使用する

・BYTE (8bit)
・WORD (16bit)
・DWORD (32bit)
・QWORD (64bit)

説明（for、while）

カウンタであるECX使用

CMPで比較

条件に満たしていればジャンプ

whileのほうがループ条件がシンプルなので分岐がシンプルになる

説明（do while）

条件に関係なく1度は処理が実行されるもの

ディスアセンブルの結果から

最初の実行をしてから繰り返していることが分かる

strings検索について

stringsコマンドを使用とghidraでも確認

コンパイルされても失われない情報を使って、プログラムの機能の推定が出来る

通信先のURLや実行するコマンドなどハードコードされていれば有用

CTFでもしばしば見かける

しかしほとんどはエンコードや難読化されている

暗号化、符号化（encrypt、encode）

符号化は公開されているアルゴリズムでエンコード、デコードする。アルゴリズムがわかれば誰でもエンコード、デコードできる。

プレーンテキストデータをある形式から別の形式に変換する

暗号化は複合化するための秘密のキー等を使用するもの。第三者からデータをみえないようにする。

（共通鍵暗号や、公開鍵暗号などと呼ばれるようなモノたち）

どちらも可逆性（もとに戻せる）

ハッシュ化

認証によく使われる（データやユーザー）パスワードやファイルの整合性

ハッシュ関数はハッシュ値というほぼランダムであるような値を出力する特殊な関数（1対1ではなく同じ値が出てしまうことを衝突という）

md5、SHA-0、SHA-1、SHA-2(SHA-256 etc...)

不可逆性（元に戻せない） 対応表を作ることで推測出来る事がある

 （←はハッシュ化するときに工夫されるもの達）

ストレッチングと呼ばれる複数回ハッシュ化を行う

ハッシュ確認してみる

windows

certutil -hashfile binary12-2.zip SHA256

linux

sha256sum binary12-2.zip

SHA256ハッシュ

2f9e3d7dcef9ec85b06b61f6f631abc51410b88ff58b4568826a00364c407ff6

XORについて

入力値A	入力値B	演算結果
1	0	1
0	1	1
1	1	0
0	0	0

排他的論理和

どちらか一方が真のとき真

どちらも真または偽のときは偽

XORファイルのlevel1をデコンパイル

説明

“Correctを表示させたい”

結果を見るとある配列にバイナリが格納されているのが分かる

0x29,0x4b,0x17,0xa,0x4b,0x49,0x25,0x2,0x4a,0x8

0x7aと入力された文字のバイナリ（ASCII文字）をXORして合っているかひとつづつ確認

全部あっていれば”Correct”と表示される

CyberChef

英国政府通信本部（GCHQ）が開発した無料のwebアプリケーション

データフォーマットや暗号化、圧縮をプログラミングせずに分析・解読することができる

https://gchq.github.io/CyberChef/

CyberChefでデコード

【レシピ名】 XORでデコード

【出典】

【材料】

From Hex
XOR:
　　Key: 0x7a (HEX)
Input:
0x29,0x4b,0x17,0x0a,0x4b,0x49,0x25,0x02,0x4a,0x08

Base64とは

バイナリデータを文字列に変換する方法で使用される文字は64種類

(a~z,A~Z)(0~9)(+,/)データのサイズを整えるのに＝が後ろにつく

データのサイズは元より増加する（35%ぐらい）

Base58などBase~はいくつかある

マルウェアではカスタムBase64という変換テーブルをカスタムしたものを使用したりする

アルゴリズム

“abc”

↓ASCIIに従ってバイナリに変換（16進数）

0x61,0x62,0x63

↓（2進数に変換）

0110 0001, 0110 0010, 0110 0011

↓連結

011000010110001001100011

↓6ビットごとに分割、最後が6ビットに満たなければ0を付与

つづき

011000 010110 001001 100011

↓変換表を用いて文字に変換

Y W J j

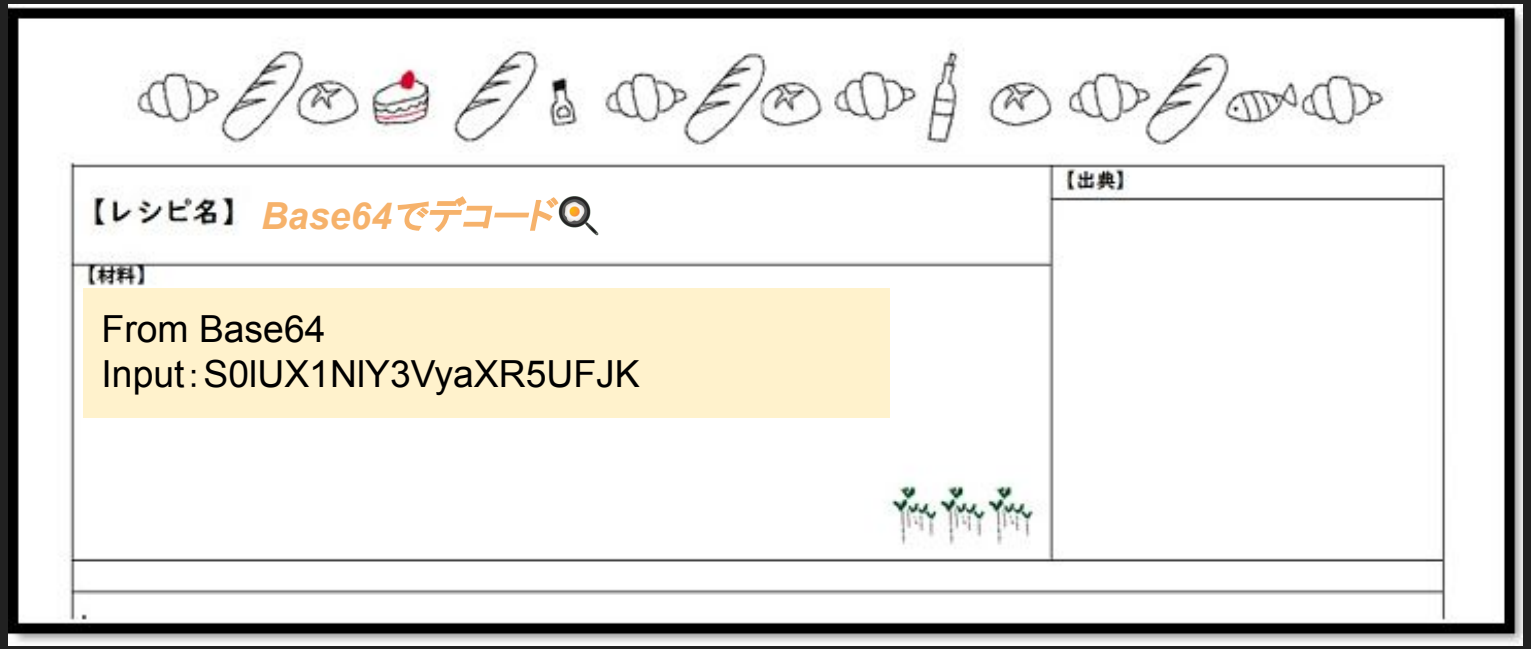
↓4文字ごとに連結最後が4文字足りなければ＝を付与

YWJj

	データ	文字		データ	文字		データ	文字		データ	文字
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/

S0IUX1NIY3VyaXR5UFJK

CyberChefでデコード



RC4について

リバースエンジニアリングされてアルゴリズムが解明された

ストリーム暗号と呼ばれる共通鍵暗号方式

マルウェアで良く使われる

詳しくはまた今度で

pcodeについて（少しだけ）

デコンパイラをするときに使用される中間言語

機械語を逆アセンブルしたものをpcodeに変換しデコンパイラされる

ghidraでpcodeを表示することも出来る

おわり