

DEFCON CTF '08 Binary 500

Analyze ELF binary in FreeBSD

Binary 500:

<http://nopsr.us/ctf2008qual/reversing500-cf3f218b2331845ae68b2a4a53b9cb28>

問題の概要(壱)

- 問題ファイルはFreeBSD用サーバプログラム
- 適度に難読化が施されている
- 適度に数学的アルゴリズムが使用されている
- 適度に音声フォーマットなどが関係している

問題の概要(弐)

- しかし、基本的に、リバーエンジニアリング以外の知識はほとんど必要ない
- 重要なのは、いかに早くアセンブルコードを読むことができるか、というその一点のみであり、ある意味、純粹に自分の解析スキルを試せる問題

解くために必要なもの

- アセンブラ(難読化含む)に関する知識
- WAVEフォーマットに関する知識
- 折れない心 (あきらめたらそこで試合終了)

Binary500問題は、純粹に**逆アセンブルコードを読む能力のみ**が試される

やるべきことは？

- 問題ファイルをgdb、IDAProで解析する
- 解析結果を元にアルゴリズム(プロトコル)を特定し、クライアントを作成する
- ihatednsサーバ上で実行されている問題プログラム(Port 2600)に対して、プロトコルに従った会話をを行い、パスワードを取得する

結論から言うと...

- Binary500の問題サーバのポート2600に connectすると、5つの数値がクライアントへ送られてくるのだが、それを周波数と考えると、その周波数の「音」を出力するWAVEファイルを生成して、サーバへ送ってやると、パスワードが返ってくるという問題

以上！

Binary500を解くまでの流れ

- ファイルタイプの特定
- 難読化方法の特定
- プロトコルの解読
- 回答を得るためのプログラムを作成

では、解析スタート！

まずは...

- ファイルタイプの特定
- 難読化方法の特定
- プロトコルの解読(推測)
- 回答を得るためのプログラムを作成

ファイルタイプの特定

- とりあえずfileしたら...

```
# file reversing500-cf3f218b2331845ae68b2a4a53b9cb28
reversing500-cf3f218b2331845ae68b2a4a53b9cb28: ELF 32-bit LSB
executable, Intel 80386, version 1 (FreeBSD), for FreeBSD 6.3, dynamically
linked (uses shared libs), stripped
```

- さらに実行したら...

```
# ./reversing500-cf3f218b2331845ae68b2a4a53b9cb28
Usage: ./MathIsHarD <keyfile>
```

...一般的な**実行ファイル**と確定！

次に...

- ファイルタイプの特定
- 難読化方法の特定
- プロトコルの解読
- 回答を得るためのプログラムを作成

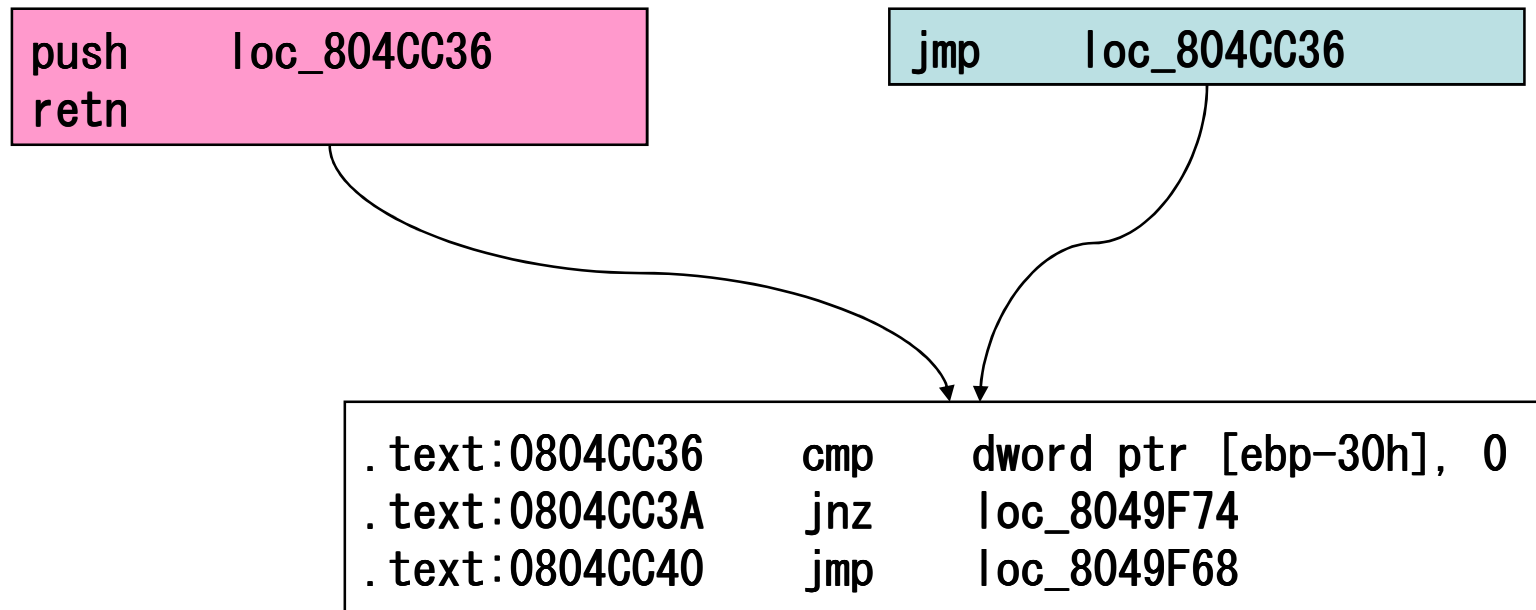
難読化方法の特定(壺)

- 難読化とは、一般的にはアセンブルコードの解析を困難にする技術
- Binary500では、コード内に、逆アセンブラ (IDAPro) が解析を誤るようなコードが、数パターン挿入されているため、まずはそれらを解読する

難読化パターン(壺)

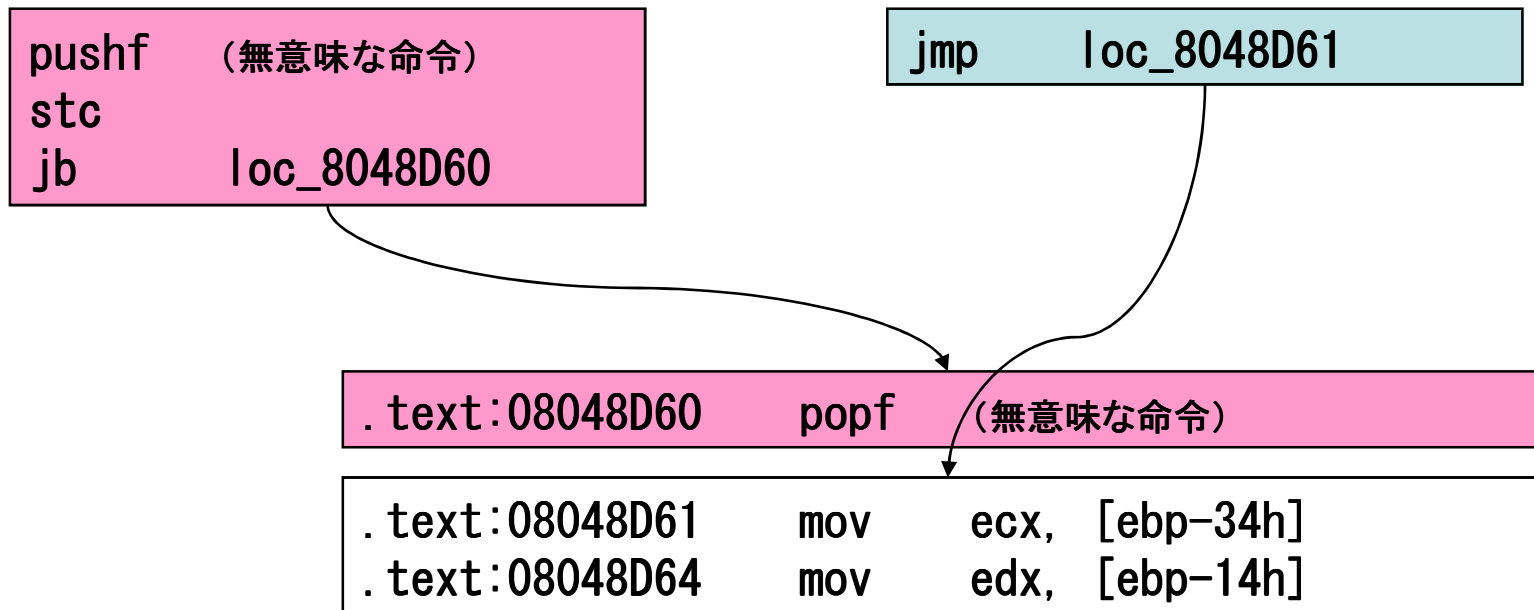
- **jmp**を**push, ret**に変換

赤と青のコードは、どちらも同じ処理(以後、同様)



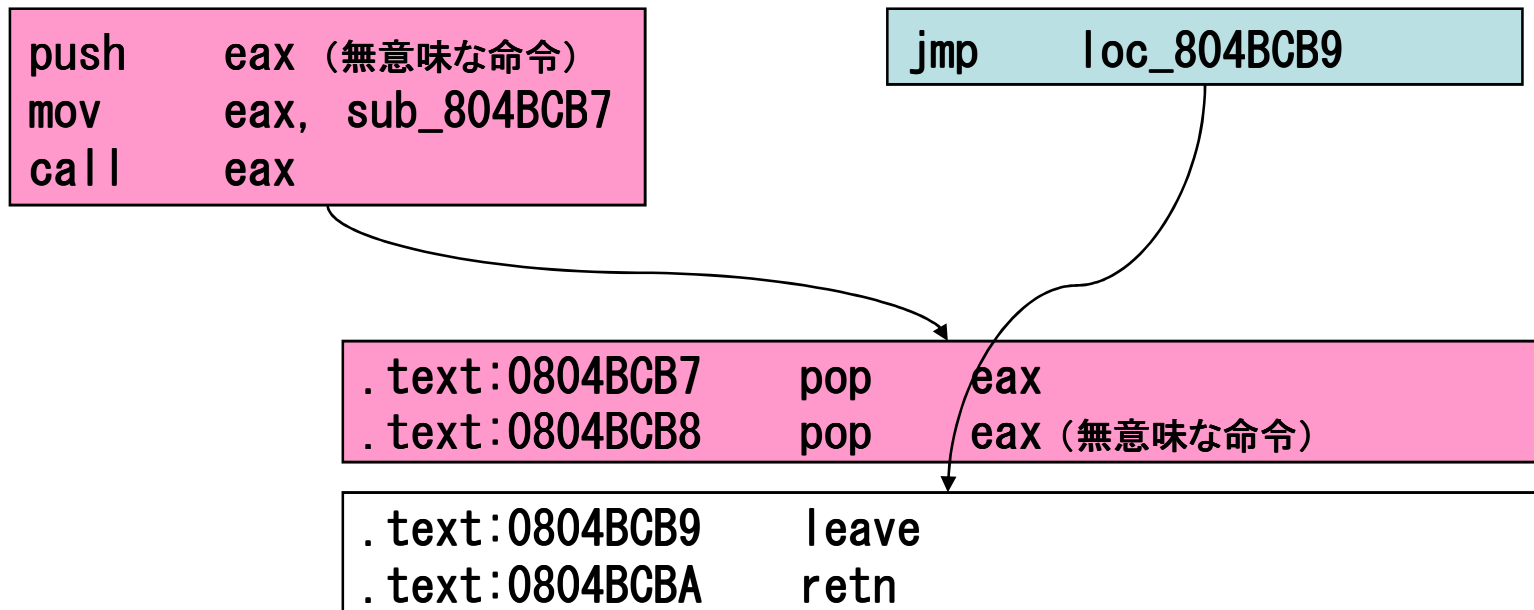
難読化パターン(弐)

- **jmp**を**stc, jb**に変換
- **無意味な命令 (pushf, popf)**を追加



難読化パターン(参)

- jmpをmov, call, popに変換
- 無意味な命令(push, pop)を追加



難読化方法の特定(弐)

- パターン数は少ない、**法則**を覚えればOK！
- ただし、IDAProの解析性能が低下することは、まぬがれない(これに関しては仕方ない)
- あとは、コード全域で使用されている難読化(以下2つ)に注意していれば、問題なし
 - 1つの関数を複数に**分割**して、**jmp命令**で**繋ぐ**
 - 関数アドレスのみを渡して、後で呼び出す

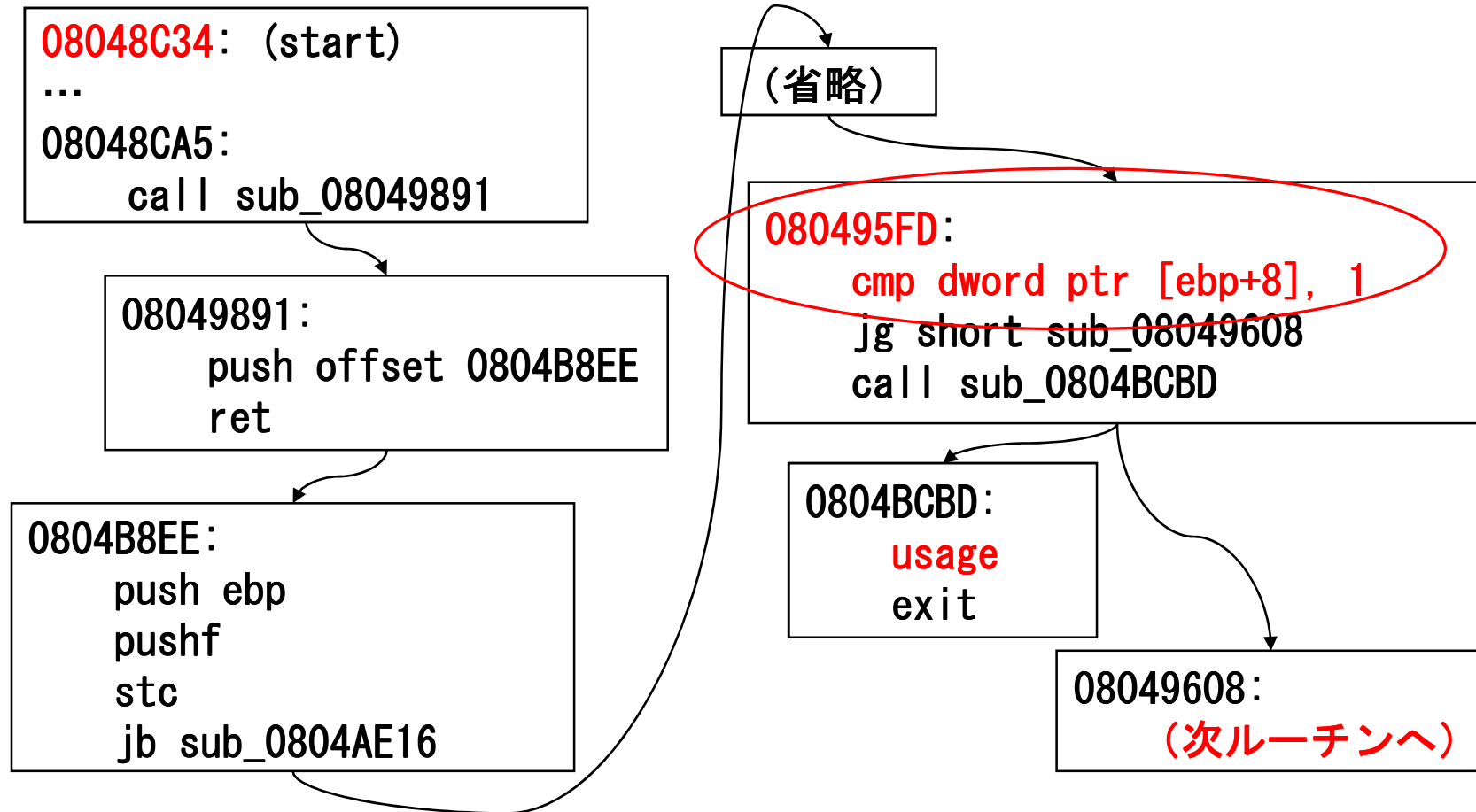
いよいよ本編へ...

- ファイルタイプの特定
- 難読化方法の特定
- **プロトコルの解読**
- 回答を得るためのプログラムを作成

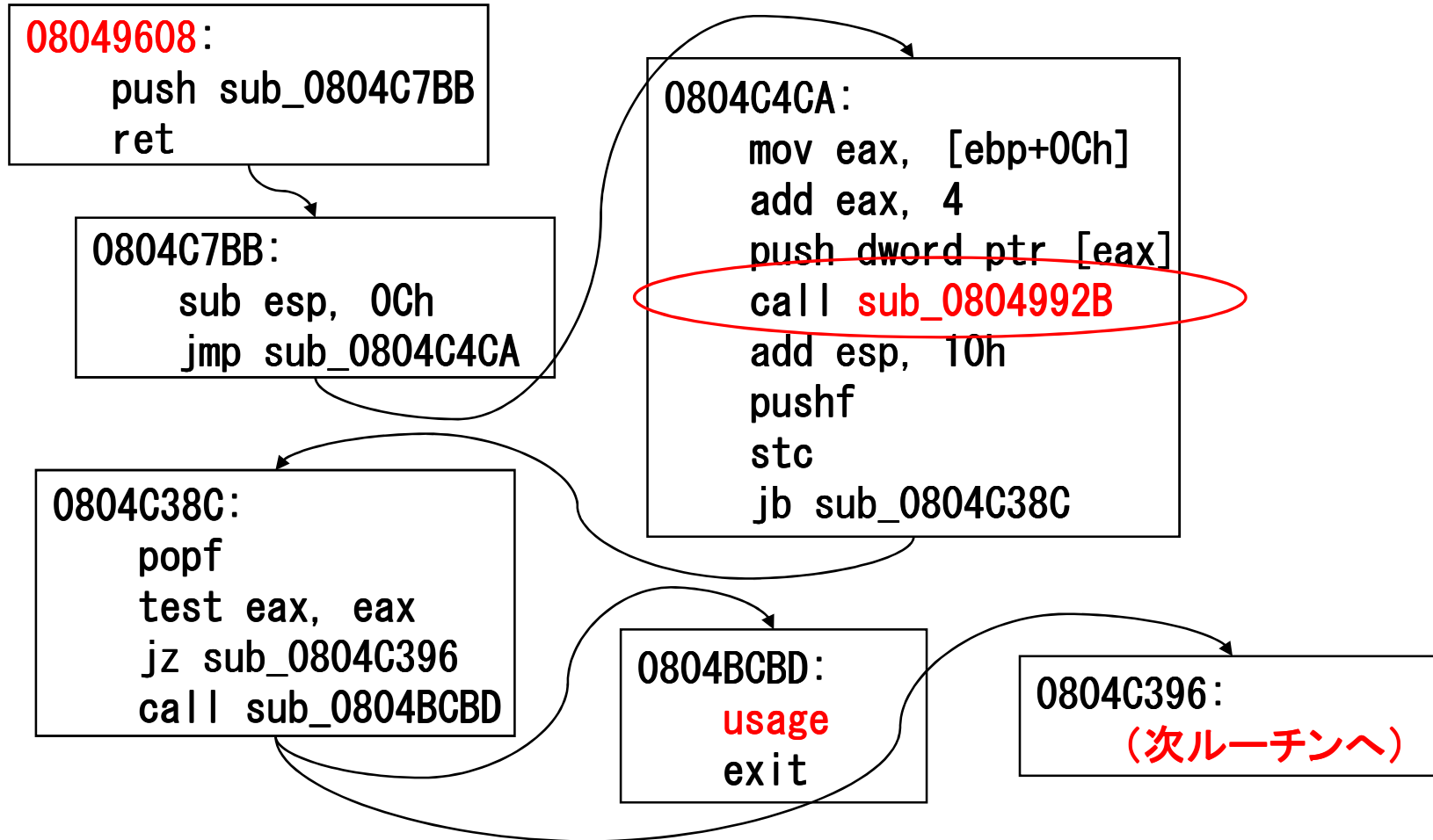
IDAProで解析

- まずはIDAProでファイルを読み込み、起動部分を解析する
- エントリーポイントは、アドレス「08048C34」であるため、ここから読み進めていく

起動時の処理(壱)



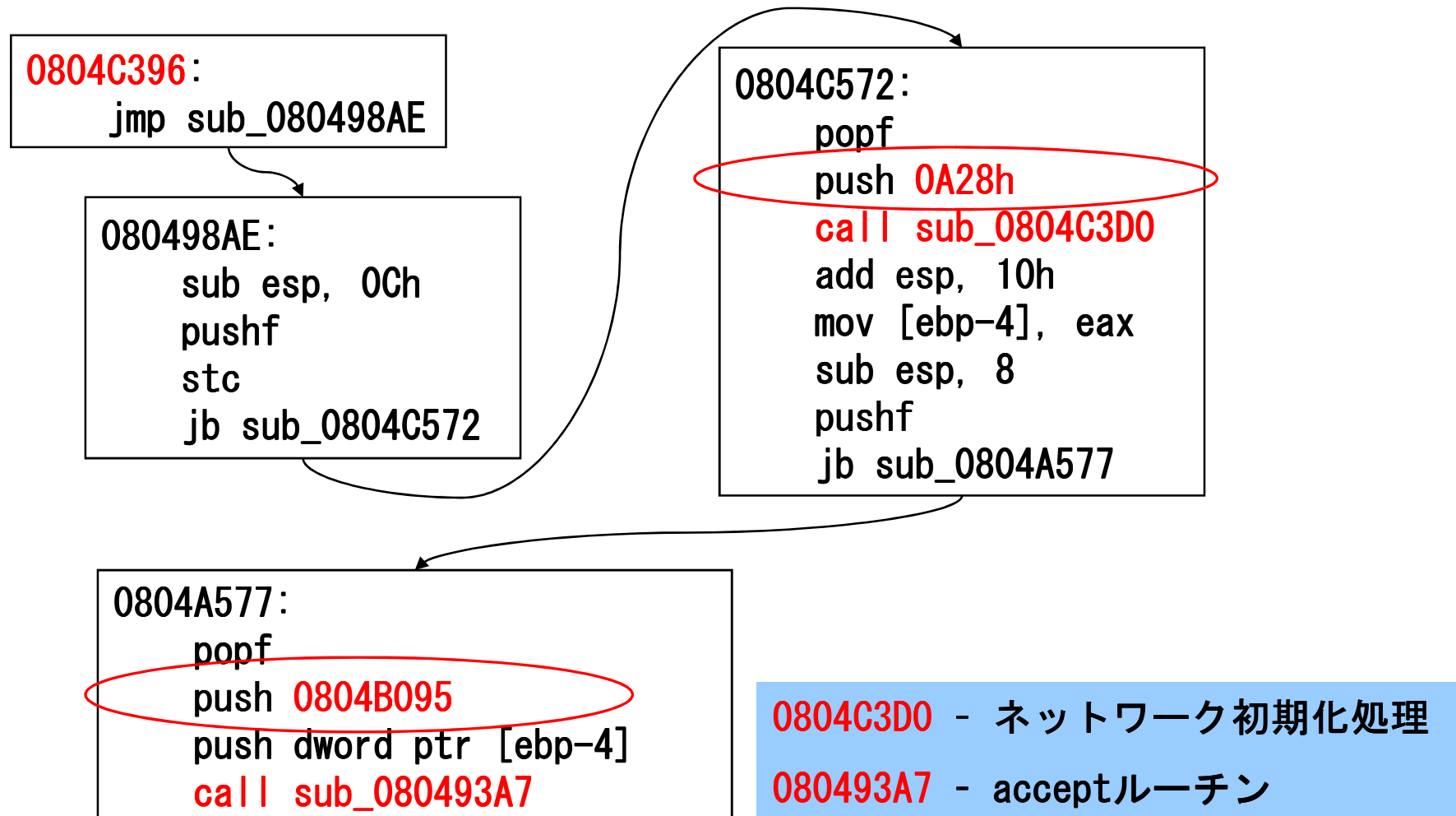
起動時の処理(弐)



起動時の処理(参)

- **080495FD**以降の処理で引数の数のチェックを行う (1より大きいならば次ルーチンへ、1以下ならばusageへ分岐する)
- **0804C4CA**以降の「**call sub_0804992B**」で飛んでいる先 (**0804992B**) では、引数に渡されたファイルを読み込む処理を行う (ちなみに、このファイルの中にパスワードが書かれてある)

サーバ初期化処理(壱)



サーバ初期化処理(弐)

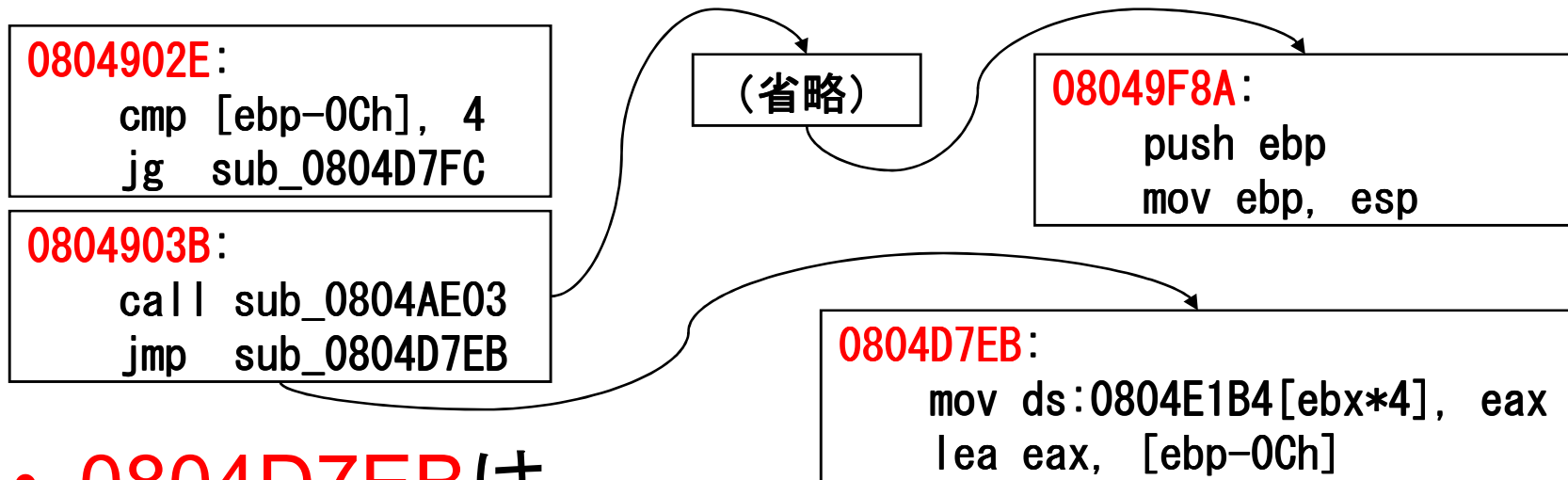
- 0804C3D0以降の処理で、ネットワーク初期化として、`socket`、`listen`などを呼んでいる(引数にポート番号0A28h(2600)を渡している)
- 080493A7以降は、`accept`ループを行う処理であり、内部で`fork`も呼んでいる(子プロセスが実行するコード(0804B095)を引数に渡している)

子プロセスの処理

- 0804B095以降が、子プロセスが行う処理であり、問題を解くために読むべき解析ルーチンとなる
- この辺りから、WAVEファイルの知識が必要になってくる

周波数生成 (08049F8A)

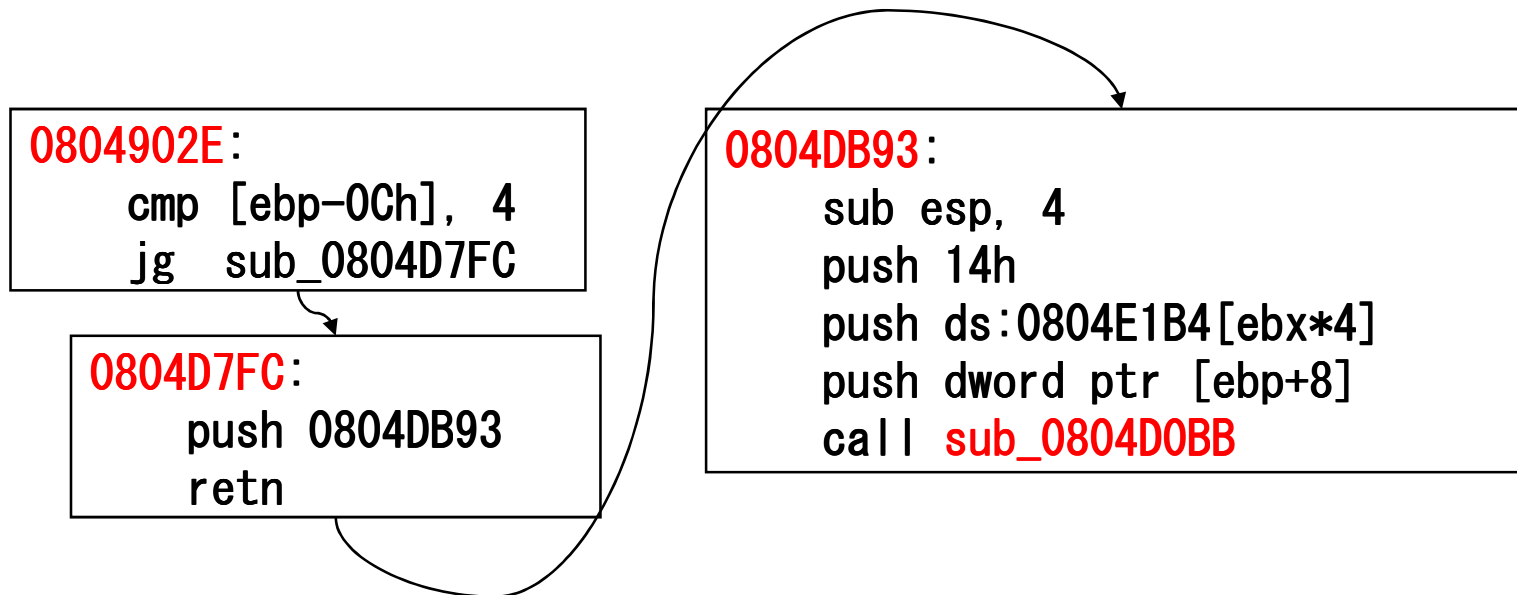
- 08049F8A以降が、クライアントへ送るべき周波数を乱数から生成する処理



- 0804D7EBは
周波数を5つ分溜める関数

周波数送信 (0804DB93)

- 0804DB93以降が周波数送信処理、sendは、実質0804D0BBで行われる



WAVEファイル受信 (0804C3E5)

- WAVEファイルは、ファイルフォーマットより、先頭から4バイト目の4バイトに、ファイルサイズが格納されているため、それを元にcallocでメモリを確保し、全データをrecvしている
- 以後の処理はWAVEファイルフォーマットを確認しながら解析した方がよい

WAVEファイルの情報源

RIFF WAVE ファイルフォーマット

<http://www.geocities.co.jp/MotorCity/3774/wave.pdf>

WAVE (.WAV) file format

<http://oku.edu.mie-u.ac.jp/~okumura/wavefmt.html>

WAV ファイルフォーマット

<http://www.kk.ij4u.or.jp/~kondo/wave/>

WAV (PCM)ファイル出力サンプル

<http://apollon.cc.u-tokyo.ac.jp/~watanabe/sample/wavsample.html>

Waveform Audio File Format (英語)

<http://groovit.disjunkt.com/analog/wave/wave.pdf>

WAVEファイルの解析処理

1. offset 00 が "RIFF" か？ (0804CE94)
2. offset 08 が "WAVE" か？ (08049B3F)
3. offset 04 が25000以下か？ (08049686)
4. offset 04 が3より大きいか？ (0804C6AD)
5. offset 04 を元にcallocを呼ぶ (0804B1D6)
6. offset 0C が "fmt" か？ (0804AA90)

このような処理が延々と続く...orz

周波数をファイルデータに変換(壱)

- サーバのチェックを突破できる**ファイルヘッダ**を作成できたら、次は周波数をファイルデータに変換するアルゴリズムを用意する
- 変換ルーチンは、問題ファイルから逆アセンブルして求めても良いが、**変換ツール**などのコードからもらってきた方が早い

周波数をファイルデータに変換(弐)

- dwWaveSizeが全サイズで、**fが周波数**、**Piが円周率**で、SAMPLING_RATEがAD変換を1秒間に何回行なうかを表す数値

```
for (int i=0; i<dwWaveSize; i++) {  
    double phase = (double) i*f*Pi*2.0 / (double) SAMPLING_RATE;  
    bWave[i] = (BYTE) (sin(phase)*128.0+128.0);  
}
```

引用元: <http://apollon.cc.u-tokyo.ac.jp/~watanabe/sample/wavsample.html>

解析結果のまとめ

- サーバに接続すると、**乱数**を元に生成される**5つの周波数**が送られてくる
- よって、それを元に**WAVEファイル**を作成し、サーバへ送り返すExploitを書く

まとめへ...

- ファイルタイプの特定
- 難読化方法の特定
- プロトコルの解読(推測)
- 回答を得るためのプログラムを作成

Exploitを作成

- 任意の周波数を元にWAVEファイルを生成するコードを作成する
- Exploitの詳細は添付されたファイル「[b500.py](#)」を参照してください

Any Question?

お疲れ様でした