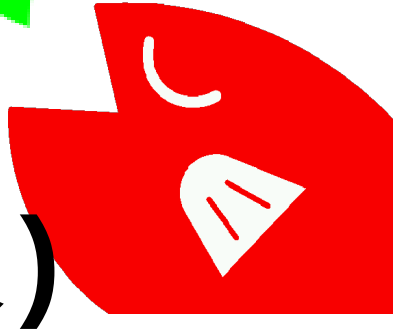
熱血シェルコード競技について

坂井弘亮
(SECCON実行委員)



ベースルール

- 競技時間は **2015年10月24日(土) 14:30-17:30** です.
- 脆弱性のあるサーバ・プログラムが多アーキテクチャ上で多数起動しています.
- サーバ・プログラムが動作しているカレント・ディレクトリに **flag.txt** という ファイルがあり, そこにフラッグ・ワードが書かれています.
- フラッグ・ワードを取得してスコアボードにサブミットすると得点になります.
- アーキテクチャは多数用意してあります. 次々と様々なアーキテクチャに挑戦 していって, 得点を重ねてください.
- 競技終了時で, 得点が最も多い競技者を優勝とします.
- 同点の場合は先にその得点に達した競技者を上位とします.

その他

- サーバに短時間に連続しての接続はできません. (数秒の間, ロックされます)
- ひとつのアーキテクチャに対して通常問題と高難易度問題が用意されています. 解く順番は自由ですが, まずは様々なアーキテクチャの通常問題を順に解いて いくことが効率的でしょう.

- (追記) サーバへの接続は数秒で自動的に切断されます.

禁止事項

- 設問によって攻略が許可されているサーバ、ネットワーク以外への攻撃
  - 他チームの端末に対する攻撃を禁止します
  - 他チームの攻略を妨害する行為を禁止します
  - その他，サーバやネットワークの動作を阻害する行為を禁止します
- 競技ネットワーク・サーバなどの負荷を過度に高める行為
  - リモートから総当たりをしないと解けない問題は、ありません
- (追記)ネットワークへの攻撃や盗聴，その他カンニングに相当する行為
- 競技ネットワークと外部ネットワークの接続
- 登録した競技者以外に、問題やヒントを公開・共有・再配布する行為
- 登録した競技者以外からの協力を得る行為
- 各競技者に提供された所定の電源以外から、電源をとること
- 許可された人以外の立ち入り
- 飲酒
- セクハラ・セキュハラ・パワハラ
- その他、運営を阻害する行為

不正行為が発見された場合、CTF競技委員長（審判）の裁量によって減点・失格などの ペナルティが競技者に対して課せられます。

競技ネットワークは
DHCPでアドレスを配布しています

競技サーバの接続先
http://192.168.1.1/

まずはブラウザで接続してみてください

トップページ

Many architectures exploit code challenge

A file named "flag.txt" are located in each server in the program's current directory.
You can obtain a flag keyword to get POINTS by reading "flag.txt".
Submit a flag keyword to SCORE BOARD.

DISTRIBUTED FILES

Easy version

Based on cross-20130826

ARM server is running on port 10000.
H8 server is running on port 10001.
MIPS server is running on port 10002.
PowerPC server is running on port 10003.
SH server is running on port 10004.
CRIS server is running on port 10005.
FR-V server is running on port 10006.
M32R server is running on port 10007.
M-CORE server is running on port 10008.
MN10300 server is running on port 10009.
SH64 server is running on port 10010.
V850 server is running on port 10011.
Thumb server is running on port 10012.
MIPS16 server is running on port 10013.

スコアサーバへのID登録

スコアボード

admin

Menu: Home History Ranking Popular Progress Logout

Number	Genre	Score	Title	Status
1	trivia	10	練習問題	
2	trivia	10	練習問題その2	
3	ARM	100	ARM (Easy)	
4	H8	100	H8 (Easy)	
5	MIPS	100	MIPS (Easy)	
6	PowerPC	100	PowerPC (Easy)	
7	SH	100	SH (Easy)	
8	CRIS	100	CRIS (Easy)	
9	FR-V	100	FR-V (Easy)	
10	M32R	100	M32R (Easy)	
11	M-CORE	100	M-CORE (Easy)	
12	MN10300	100	MN10300 (Easy)	
13	SH64	100	SH64 (Easy)	
14	V850	100	V850 (Easy)	
15	Thumb	100	Thumb (Easy)	
16	MIPS16	100	MIPS16 (Easy)	
17	Blackfin	100	Blackfin (Easy)	

競技サーバへの接続

```
$ telnet 192.168.1.1 10011
This is v850-elf server.
Push Enter:
OK.
Input name:
OK. Your name:
$
```

- 様々なアーキテクチャの競技サーバが動作しています.
- 各競技サーバは, GDBのシミュレータで動作しています.
- GDBシミュレータで出力がバッファリングされるため，出力が遅れる場合があります. (競技には影響ありません)

競技サーバの攻略

- 競技サーバの実行ファイルは, 競技のトップページからダウンロードできます.
  (V850の実行ファイル**:v850-elf.x)**

- 脆弱性があります. 脆弱性を利用して, サーバ上のファイルの内容を読み取る ような**Exploit**が可能です.

- **Exploit**コードを作成し, サーバに送り込み, カレントディレクトリにある **flag.txt** の内容**(**フラッグワード**)**を読み取ってください.

- フラッグワードをスコアサーバにサブミットすると, 得点になります.

攻略のために(1)

- サーバは以下のクロスコンパイル環境をベースにして作成しています.

  http://kozos.jp/books/asm/
  cross-20130826.zip / cross-gcc4-20130826.zip

- サーバは上記環境で構築したGDBのシミュレータで動作しています.
  GDBは素のままで, 改造無しで使っています.

攻略のために(2)

- **cross-20130826.zip / cross-gcc4-20130826.zip** をインストールすると，様々なアーキテクチャのクロスコンパイル環境が使えるようになります．
- ただしインストールにものすごく時間がかかるので，インストール済みの VM イメージが以下で配布されています．

  http://kozos.jp/vmimage/

- **cross-20130826.zip / cross-gcc4-20130826.zip** を展開すると**exec** という ディレクトリがあり，各アーキテクチャのサンプルプログラムを**GDB**シミュレータで 動作させることができます．この環境を利用してサーバを動作させています．
- これらの必要ファイルは，スコアサーバからダウンロードできます．

攻略のために(3)

- まずは実行ファイルを逆アセンブルしてみましょう.

  (V850の場合)
  ```
  $ /usr/local/cross/bin/v850-elf-objdump -d v850-elf.x
  ```

- **GDB**でデバッグができるアーキテクチャもあります.

  ```
  $ /usr/local/cross/bin/v850-elf-gdb v850-elf.x
  (gdb) target sim
  (gdb) load
  (gdb) break main
  (gdb) run
  (gdb) layout asm
  ```

- **GDB**シミュレータを利用して,実行ファイルを手元で動作させることもできます. トレースなど出力させてみるといいかもしれません.

  ```
  $ /usr/local/cross/bin/v850-elf-run v850-elf.x
  $ /usr/local/cross-gcc4/bin/rx-elf-run rx-elf.x
  ```

攻略のために(4)

- 各アーキテクチャのアセンブラを理解するために, クロスコンパイル環境が利用できます.
  cross-20130826 / cross-gcc4-20130826 の **sample** のディレクトリを参考に, フィーリングで読んでください.

- 各アーキテクチャの機械語コードを生成するために, クロスコンパイル環境が 利用できます.
  cross-20130826 / cross-gcc4-20130826 の **exec** のディレクトリに 各種アーキテクチャのスタートアップとシステムコール発行の例が ありますので参考にしてください.

- 競技サーバのプログラムは, 同じ**C**言語ソースコードからアーキごとにビルド したものです. どれかひとつのアーキで解析できれば, 他のアーキも動作が推測できるでしょう.

攻略のために(5)

- **flag.txt**の読み取りのためには，システムコールの発行が必要です．システムコールの発行方法は，**GDB**シミュレータのソースコードを見てください．
- **GDB**のソースコードを解凍すると，**sim**というディレクトリがあります．そこがシミュレータのソースコードです．
- アーキテクチャごとの細かい動作を知りたいときにも，**GDB**シミュレータのソースを読むといいでしょう．
- **GDB**は素のままで，改造無しで使っています．**GDB**独自の動作や，他ドキュメント等との仕様の食い違いや，そもそもバグなどが あるかもしれませんが，それらを読み取るのも競技のうちと思ってください．

攻略のために(6)

- 競技サーバは最初に入力を受けて, さらに名前の入力を受けます.
- スクリプトなどでたて続けに入力データを送ると, ひとつの**read()**でまとめて 受信してしまい, **Exploit**が期待通りに動かなかったりします.
- 最初の入力と次の入力の間にタイムラグを持たせると回避できます.
- これを行うための**cat.pl**というスクリプトがダウンロードできますので 参考にしてください.

```
$ ./cat.pl enter.bin exploit.bin enter.bin ¥
        | nc 192.168.1.1 10000
```

- 標準入力を待って, 入力待ちを入れるとうまくいく場合もあります.

```
$ ./cat.pl enter.bin exploit.bin enter.bin - ¥
        | nc 192.168.1.1 10000
```

攻略のために(7)

- 出力が適宜フラッシュされず，入力後に遅れて出力がされるような場合がありますが，競技に影響は無いので気にしないでください．

- GDBシミュレータは使えるがGDBデバッガがうまく使えなかったり，そもそも標準環境ではビルドできなかったりするアーキテクチャもあります．それをなんとかするのも競技のうちと思ってください．

- 通常問題と高難易度問題があります．
  得られるスコアは同じなので，まずは通常問題を解いていくのがいいでしょう．高難易度問題は，入力できる文字に制限があります．

では，がんばってください！