セキュリティ・キャンプ全国大会 2015 応募用紙

ふりがな　たむら　だい
氏 名　田村　大
電話番号(昼間に連絡できる電話番号)
E-mail アドレス(必ず書いてください)
所 属(学校・学科・学年) 岡山大学工学部情報系学科 4 年生
年齢(応募現在)　（ 21 ）才　（※2016 年 3 月 31 日時点において 22 歳以下の学生・生徒であることが応募条件です）
Twitter アカウント(ある場合のみ)
ホームページまたはブログの URL(ある場合のみ)

共通問題

下記の共通問題にはすべて回答してください。

■ 共通問題 1

セキュリティ・キャンプに応募した自分なりの理由とセキュリティ・キャンプで学んだことを何に役立てたいかを教えてください。

【以下に回答してください（行は適宜追加してください）】

私は現在大学の所属研究室でオペレーティングシステムの研究をしている．今までセキュリティに対しての知識を得る機会はあまりなく，興味もあまりなかった．しかし，研究室の先輩がセキュリティの研究をしており，研究の話を聞かせてもらい，セキュリティに興味がわいた．セキュリティ・キャンプに応募できる機会は今年で最後なので，いい機会だと思い応募した．セキュリティ・キャンプで得た経験とセキュリティの知識からセキュリティに対する更なる興味を持ち，同時に自身の研究へ役立てたい．

【回答ここまで】

■ 共通問題 2

セキュリティに関することで、過去に自分が経験したことや、ニュースなどで知ったことの中から、最も印象に残っていることを教えてください。また、その印象に残った理由も教えてください。

【以下に回答してください（行は適宜追加してください）】

私が直接経験したことではなく友人が経験したことであるが，最も印象に残っていることは友人の計算機が「仁義なきキンタマ」という名のトロイの木馬型ワームに感染したことだ．当時はまだ幼く，話を聞いただけなのだが，名前に衝撃を感じ，今でも印象に残っている．

【回答ここまで】

■ 共通問題 3

その他に自己アピールしたいことがあれば自由に書いてください。（たとえば、あなたが希望する講座を受講する上で、どのような技術力を持っているか、部活動、技術ブログ、GitHub、ソフトウェア開発、プログラミングコンテスト、勉強会での発表・運営などの実績や熱意があれば、あるだけ書いてください。）

【以下に回答してください（行は適宜追加してください）】

バレーボールを中学校から大学生である今現在までやっており，チームプレイが得意である．グループワークなどではチームの士気を高めることができると思う．

また，技術的な面では使用できる言語は C 言語のみであり，アセンブラは MIPS アーキテクチャと x86 アーキテクチャを勉強した．4 月から研究室に所属し始め，FreeBSD4.3 のカーネルのビルドやシステムコールの作成などを行った．現在は研究テーマに関する調査を行っている．

また現在，個人的にオペレーティングシステムを作成中である．

【回答ここまで】

【ページ区切りはこのままにしてください。】

選択問題

下記の選択問題の中から 5 つ 選択して回答してください。
なお、回答した問題は、冒頭の□を■にしてください。

■選択問題 1 （左側の□について、回答した問題は■にしてください）

過去に家電製品や電子機器を分解して中身を見てみたことはありますか？（ある場合のみ回答）
(1) もしもあれば何を分解したのか具体的な製品名を添えて 2 個以上教えてください。
(2) 何の目的のためにそれを分解しようと思ったのでしょうか。理由を教えてください。
(3) 分解してみて中に何があったのか、わかったこと、自分で発見できたことを書いてください。
(4) 分解してみてもわからなかったこと、今後勉強してみたいと思っている内容を書いてください。
【以下に回答してください（行は適宜追加してください）】
(1) ゲームボーイアドバンスのゲームソフト. 製品名は「ポケットモンスター　ルビー」
(2) ゲームソフト内の配線のどこかを切れば新しいポケモンが手に入るというデマが流行り，自分も実際に配線を切ってみようと考えたから.
(3) 実際に配線を何か所か切ってみた. ゲームが起動しないかと思ったが普通に起動し驚いた. と同時に本当に新しいポケモンが出現するのではないかと期待した. 結果は切断前と変わらないゲームの内容であった. がっかりはしたものの，配線を何本か切ってもゲームが起動できることには驚いた.
(4) 分解してみてもどの配線がどのような役割を果たしているのか全く分からなかった. 切断しても無事動くならばその配線はいらないのではないかという疑問が生じた. どのような理由で配線されているのか知りたい.
【回答ここまで】

■選択問題 2 （左側の□について、回答した問題は■にしてください）

ある機械語を objdump により逆アセンブルしたところ、以下の結果が得られました。このダンプに見られる問題点を指摘してください。

```
00000000 <.text>:
   0:   b8 de 07 00 00          mov    $0x7de,%eax
   5:   3d df 07 00 00          cmp    $0x7df,%eax
   a:   75 06                   jne    0x12
   c:   89 c0                   mov    %eax,%eax
   e:   ff e3                   jmp    *%ebx
  10:   75 01                   jne    0x13
  12:   e9 5b ba 0e 00          jmp    0xeba72
  17:   00 00                   add    %al,(%eax)
  19:   b9 00 00 00 00          mov    $0x0,%ecx
  1e:   bb 01 00 00 00          mov    $0x1,%ebx
  23:   b8 04 00 00 00          mov    $0x4,%eax
  28:   cd 80                   int    $0x80
  2a:   b8 01 00 00 00          mov    $0x1,%eax
  2f:   cd 80                   int    $0x80
```

【以下に回答してください(行は適宜追加してください)】

Linux32bit の機械語コードであると仮定する. なぜならば 0x28 番地からの命令である"int $0x80"は Linux32bit の機械語コードにおいてシステムコールの呼び出し命令だからである. 以下にこのプログラムがどこから始まるかを場合分けし, それぞれのプログラムの動きを解説する.

(1) このプログラムの 0x0 番地から実行されると仮定する場合
この場合, 0x5 番地の cmp 命令の解は明らかに偽であるため, 0xa 番地のジャンプにより必ず 0x12 番地の命令が実行される. 0x12 番地の jmp 命令により 0xeba72 番地へジャンプするが, たいていの場合 0xeba 番地には実行権限がないと考えられるため, セグメンテーションフォールトとなることが問題点である.

(2) このプログラムが 0x10 番地から実行されると仮定する場合
この場合, 0x10 番地から 0x13 番地にジャンプする. 0x13 番地と 0x14 番地～0x18 番地をそれぞれ"pop %ebx"と"mov $0xe,%edx"という命令に解釈することができる. この時, プログラムの挙動としては, 0x14 番地～0x27 番地で edx, ecx, ebx, eax レジスタにそれぞれ 0xe, 0x0, 0x1, 0x4 の値を格納している. 0x23 番地で eax レジスタに 0x4 が格納されたことから, 0x28 番地で write システムコールが実行されている. write システムコールは edx, ecx, ebx レジスタの値から, NULL が指す番地から 14 バイト分を標準エラー出力する. その後, 0x2a 番地～0x30 番地で exit システムコールを実行し, プログラムを終了する. ここで問題点となることは ecx レジスタの値が 0x0 であるという点である. つまり, write システムコールは NULL を参照している点である. 仮定した本環境においては 0x0 番地はページが存在せず, 読み込むことができないからである.
実際に C 言語内に上記のアセンブリコードを入力して実行してみると, wirite()システムコールで EINVAL 引数エラーとなった. そのため, 何も出力されなかった.

(3) このプログラムが 0xc 番地または 0xe 番地から実行されると仮定する場合
この場合，0xc 番地の命令は特に意味をなさない命令であり，0xe 番地で ebx レジスタに格納されているアドレスにジャンプする．ここでの問題点は ebx レジスタに格納されている値が不明であるという点である．ebx レジスタに格納されている値が適切な値の場合，問題はない．しかし，不適切な値の場合，セグメンテーションフォールトになる．

【回答ここまで】

□選択問題 3 （左側の□について、回答した問題は■にしてください）

過去に作成したプログラムのうち最も気に入っているものについて答えてください。
ここでいうプログラムは、Web サービスやモバイルアプリ、サーバ/デスクトップアプリケーションあるいは OS、VM などといったソフトウェア全般のことです。
(1)どのようなソフトウェアであるかを教えてください
(2)何の目的のためにそれを作ろうと思ったのか、理由を教えてください
(3)開発するにあたって工夫したところを教えてください
(4)新たな課題、今後勉強してみたいと思っている内容を書いてください

【以下に回答してください（行は適宜追加してください）】

【回答ここまで】

□選択問題 4 （左側の□について、回答した問題は■にしてください）

通信やプロトコルに関する問題です。次の 1-3 の問いに回答してください。ただし 3 は任意とします。
(1)インターネット上で自分が興味を持ったサイト(又はホスト)を見つけ、自分の端末からそのサイトへ ping を打ちなさい。その結果とそのサイトに興味を持った理由を記述してください
(2)自分の端末から 1 のサイトの間のネットワーク通信が、技術的にどのような仕組みで何が行われているのか考察して記述してください（使用する通信は ping に限定しません）
(3)可能であれば、自分の端末から 1 のサイトの間で、どのようなプロトコルを使ってどういったサービスが実現できているのか、また将来どんな事が実現できれば良いだろうか、考察して記述してください

【以下に回答してください（行は適宜追加してください）】

【回答ここまで】

■ 選択問題 5 （左側の□について、回答した問題は■にしてください）

以下のような C 言語の関数 function があるとします。

```
void function(int *array, int n) {
  int i;
  for(i = 0; i < n; i++) {
    array[i] = i * n;
  }
}
```

上記プログラムをコンパイルした結果の一例 (i386)は以下となりました。

```
00000000 <function>:
   0:   56                      push   %esi
   1:   53                      push   %ebx
   2:   8b 5c 24 0c             mov    0xc(%esp),%ebx
   6:   8b 4c 24 10             mov    0x10(%esp),%ecx
   a:   85 c9                   test   %ecx,%ecx
   c:   7e 18                   jle    26 <function+0x26>
   e:   89 ce                   mov    %ecx,%esi
  10:   ba 00 00 00 00          mov    $0x0,%edx
  15:   b8 00 00 00 00          mov    $0x0,%eax
  1a:   89 14 83                mov    %edx,(%ebx,%eax,4)
  1d:   83 c0 01                add    $0x1,%eax
  20:   01 f2                   add    %esi,%edx
  22:   39 c8                   cmp    %ecx,%eax
  24:   75 f4                   jne    1a <function+0x1a>
  26:   5b                      pop    %ebx
  27:   5e                      pop    %esi
  28:   c3                      ret
```

このとき以下の(1)～(5)の設問について、回答と好きなだけ深い考察を記述してください。知らない点は、調査したり自分で想像して書いてもらっても結構です。どうしてもわからない部分は、具体的にここがわかりませんと記述しても良いです。(1)～(2)の回答は必ず答えてください。(3)～(5)の回答は任意です。わかることを書いてください。CPU やコンパイラは特定の実装を例に説明しても良いですし、理想を自由に考えても良いです。

(1)【必須】上記の C 言語のプログラムはどのような動作をしますか。また、この関数を呼び出して利用する main 関数の例を作成してください。
(2)【必須】上記のアセンブリコードを、いくつかのブロックに分割して、おおまかに何をしている部分かを説明してください。もし、上記のアセンブリが気に入らないのであれば、好きなアーキテクチャやコンパイラのアセンブル結果を載せて説明しても良いです。
(3)【任意】コンパイラがソースコードの関数を解釈して、ターゲットのアーキテクチャのバイナリを生成するまで、どのように内部で処理を行っていると思いますか。(キーワード: 構文解析、変数、引数、呼出規約、レジスタ、スタック、アセンブラ、命令セット)
(4)【任意】CPU の内部では、プログラムのバイナリはどのように解釈され実行されていると思いますか。(キーワード: フェッチ、デコード、オペコード、オペランド、命令パイプライン、回路)
(5)【任意】現在の CPU やコンパイラの不満点があれば自由に記述してください。

【以下に回答してください(行は適宜追加してください)】

(1) まず上記の C 言語プログラムの動作を解説する. このプログラムは引数として int 型のポインタ array と int 型の変数 n を受け取る. そして, for ループにより, array が示すアドレスに i x n の値を格納し, その 4 バイト先のアドレス, その 8 バイト先のアドレスというふうに n 回 I x n の値を格納する処理を行う. 以下に function 関数を呼び出す main 関数の例を示し, 解説する.

```
/*九九の表を作成し出力*/
int main(void){

  int i,j,n=9+1;        /*n は九九の段プラス 1 の数*/
   int array[n][n];     /*九九の表を格納する配列*/

  /*function 関数により表の半分を作成*/
  for(i=0;i<n;i++){
    function(&array[i][0],i);
    array[i][i]=i*i;
  }

  /*表の残り半分を埋め出力
    0 の段は非表示              */
  for(i=1;i<n;i++){
    for(j=1;j<n;j++){
      array[i][j]=array[j][i];
      printf("%2d ",array[i][j]);
    }
    printf("¥n");
  }

  return 0;
}
```

この main 関数は function 関数により九九の表を作成し，標準出力するプログラムである．
functioin 関数では九九の表は半分しか作成することができないが，九九の対称性を使うことにより九九の表を作成できる．ここでいう対称性とは n x m = m x n (n, m は 1 から 9 の整数)という性質である．以下に実行結果を示す．

```
$ ./a.out
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81

$
```

(2) 上記のアセンブリコードを，いくつかのブロックに分割し，それぞれの部分を説明したものを以下の表にまとめる．

アドレス	説明
0x0～0x1 番地	レジスタの退避．
0x2～0x9 番地	引数をそれぞれレジスタに格納．
0xa～0xd 番地	for 文における初期化式が継続条件式に当てはまるかを確認．当てはまらない場合は 0x26 番地へジャンプ．
0xe～0x19 番地	ループの際に使うレジスタの初期化．
0x1a～0x1c 番地	ポインタが指すアドレスに i x n の計算結果の値を格納．
0x1d～0x21 番地	再初期化式を実行．
0x22～0x24 番地	継続条件式を確認し，ループを継続する場合 0x1a 番地にジャンプ
0x26～0x27 番地	レジスタの値を function 関数が呼び出された時の状態に戻す．
0x28 番地	function 関数を呼び出したアドレスへジャンプ．

(3) 私は C 言語に類似した文法のコンパイラを作成したことがある．そのコンパイラをもとに解説をする．まずコンパイラはソースコードを字句解析する．字句解析によりソースコードを構成する文字列を字句に分解する．その後，構文解析を行う．字句解析により分けられた字句列から計算機に対して構文木を生成し，すべての字句の並びが定義された文法に従っているかを判定する．その後コード生成にて，特定の命令セットで記述されたファイルを出力する．コード生成をする際，ローカル変数はスタックで管理され，グローバル変数やスタティック変数はデータ部で管理される．引数や返り値は呼び出し規約によりどのように値を渡すかを決める．具体的な例としては，引数はスタックに順に値を積んで渡し，返り値は特定のレジスタに値を格納する．
アセンブラはコンパイラによって作成されたファイルに記述されている命令セットをもとにバイナリを生成する．

(4) CPU は命令バイナリを 4 つのステージ(命令フェッチ，命令デコード，実行，ライトバック)で一つの命令を処理する．まず命令フェッチステージでは命令コードをメモリから読み込み，プロセッサ内部のレジスタに格納する．次に命令デコードステージでレジスタに格納された命令コードを解析し，実

行の準備を行う. 解析はオペコードにより, 命令を対応付け, オペランドでレジスタやアドレス番地を指定する. その次に実行ステージでは CPU 内部の演算ユニットで命令を実行する. 最後のライトバックステージで実行結果をレジスタやメモリに書き込む. この4つのステージが回路レベルで独立していることにより, 命令の処理を独立できる. これにより, 流れ作業的に、前の命令のサイクルが完了する前に次の命令を処理し始めることができ, プロセッサの処理の高速化が可能となる.

(5) 現在の CPU に対する不満点は発熱量である. 私が現在使用しているノートパソコンの CPU は Intel Core i7-2670QM 2.20Ghz (TDP:45W)であるが, 発熱がとても気になる. アイドル時、CPU の温度は 45 度くらいであるが, CPU 負荷テストのツールである prime95 を使用した際には 90 度を超えた. Intel は CPU の性能を上げる際, クロック周波数を増やすことで性能を上げる傾向にあると聞いたことがある. 今現在最新の CPU がどれくらいの発熱量かは知らないが, Intel の傾向であると発熱量が少なくなっているとは考えにくい. 昔使っていたノートパソコンの CPU は Intel Core i3-2377M 1.50Ghz (TDP:17W)であったが, 発熱量を気にした記憶はない. これからの CPU には, TDP を 20W 代に維持した状態で今現在の性能を維持できるようになってもらいたい. それにより発熱が気にならなくなることを期待している.

現在のコンパイラに対する不満点は処理系の不完全性である. Standard ML 言語において文法上では正しいにもかかわらず, 処理系がエラーを示すことがあった. 処理系は文法すべてに対応してから世に出回ってほしいものである.

【回答ここまで】

□選択問題 6 （左側の□について、回答した問題は■にしてください）

Linux システムの問題解決であなたが使ったことのあるツール（例： strace, gcore, gdb, kdump など）について記述してください（差支えの無い範囲で構わないので、問題の内容／使用したツール／どのように使用して解決したのかをなるべく詳しく説明 してください）。

【以下に回答してください（行は適宜追加してください）】

【回答ここまで】

□選択問題 7 （左側の□について、回答した問題は■にしてください）

あなたのネットワークに不審な通信を行っている PC があります。この PC を特定し、中身を調査するためにとるべきアプローチについて具体例を挙げて熱烈にアピールしてください。

【以下に回答してください（行は適宜追加してください）】

【回答ここまで】

■選択問題 8 （左側の口について、回答した問題は■にしてください）

gcc が持つ-fno-stack-protector は、どのようなセキュリティ機能を無効にするオプションであるのか、またこの機能により、どういった脆弱性からソフトウェアを守れるのかをそれぞれ記述してください。

【以下に回答してください（行は適宜追加してください）】

“-fno-stack-protector”オプションはスタックの中の関数リターンアドレスの付近などでの不適当な値の書き変えを検知する機能を無効にするオプションである．具体的には、4 バイトの「カナリア」と呼ばれるワードが配置され，関数からのリターン直前にカナリア値が書き変わっていないかがチェックすることで，値の書き換えがないかをチェックしている．

実際に C 言語でプログラムを作成し，objdump することで，カナリア値の存在を確認した．以下で作成したプログラムのソースコード，objdump の結果，および解説を示す．使用した GCC は ver.4.2.1 であったため，デフォルトではカナリア値は設定されなかった．そのためカナリア値を確認するために fstack-protector-all のオプションを使用した．

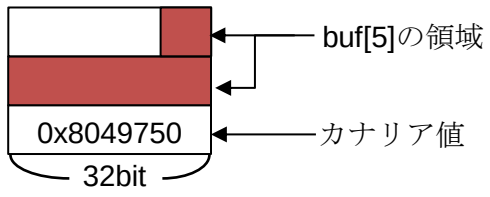
（実行環境　OS：FreeBSD8.2-RELEASE　CPU:Intel Core(TM) i7-4790S(4.00GHz)

```
/*カナリア値を確認するために作成したプログラムのソースコード*/
/*q8.c*/
#include<string.h>

int main(int argc,char *argv[]){
        char buf[5];

        strcpy(buf,argv[1]);
        return 0;
}
```

#dump したプログラムの一部

```
080484b0 <main>:
 80484b0:       8d 4c 24 04             lea    0x4(%esp),%ecx
 80484b4:       83 e4 f0                and    $0xfffffff0,%esp
 80484b7:       ff 71 fc                pushl  0xfffffffc(%ecx)
 80484ba:       55                      push   %ebp
 80484bb:       89 e5                   mov    %esp,%ebp
 80484bd:       51                      push   %ecx
 80484be:       83 ec 24                sub    $0x24,%esp
 80484c1:       8b 01                   mov    (%ecx),%eax
 80484c3:       89 45 e8                mov    %eax,0xffffffe8(%ebp)
 80484c6:       8b 41 04                mov    0x4(%ecx),%eax
 80484c9:       89 45 e4                mov    %eax,0xffffffe4(%ebp)
 80484cc:       a1 50 97 04 08          mov    0x8049750,%eax          #カナリア値をスタックに積む
 80484d1:       89 45 f8                mov    %eax,0xfffffff8(%ebp)
 80484d4:       31 c0                   xor    %eax,%eax
 80484d6:       8b 45 e4                mov    0xffffffe4(%ebp),%eax
 80484d9:       83 c0 04                add    $0x4,%eax
 80484dc:       8b 00                   mov    (%eax),%eax
 80484de:       89 44 24 04             mov    %eax,0x4(%esp)
 80484e2:       8d 45 f3                lea    0xfffffff3(%ebp),%eax   #カナリア値を格納したスタックのアドレスのすぐ上位のアドレ
                                                                       #ス番地に buf[5]の領域確保
 80484e5:       89 04 24                mov    %eax,(%esp)
 80484e8:       e8 8f fe ff ff          call   804837c <_init+0x64>    #strcmp()の呼び出し
 80484ed:       b8 00 00 00 00          mov    $0x0,%eax
 80484f2:       8b 55 f8                mov    0xfffffff8(%ebp),%edx
 80484f5:       33 15 50 97 04 08       xor    0x8049750,%edx          #カナリア値が書き換えられていないかの確認
 80484fb:       74 05                   je     8048502 <main+0x52>     #カナリア値が書き換えられていない場合ジャンプし処理を続ける
 80484fd:       e8 5a fe ff ff          call   804835c <_init+0x44>    #カナリア値が書き換えられている場合

                        <略>
```

スタックの解説図



プログラムの実行例

```
$gcc -fno-stack-protector q8.c
$./a.out aaa
$./a.out aaaaaaaa
Segmentation fault (core dumped)
$gcc -fstack-protector-all q8.c
$./a.out aaa
$./a.out aaaaaaaa
allprotect: stack  overflow detected;terminated
Abort (core dumped)
```

objdump の結果より，0x80484cc 番地でスタックにカナリア値を格納し，0x80484f5 番地でカナリア値が書き換えられていないかを確認していることが分かる．カナリア値が書き換えられていない場合は処理を続行し，書き換えられている場合はエラー処理へと移行する．カナリア値は buf の値を格納するために確保されているスタック領域の直後の場所に格納されている．これにより，決められたバッファの大きさ以上のデータが格納されたかどうかを検知することができる．
このような不適切な値の書き換えを検知する機能が有効になっていることで return アドレスなどの改竄を検知し，改竄を確認した場合プログラムを終了させることにより，スタックスマッシング攻撃によるバッファオーバーフローなどの脆弱性からソフトウェアを守ることができる．
そこで，バッファオーバーフローによる脆弱性を持つプログラムの一例を作成した．このプログラムは第 1 引数が文字列”clalis”と一致するかどうかを確認し，一致する場合は”correct password”，一致しない場合は”wrong password”と標準出力するプログラムである．以下にプログラムのソースコードと実行例を記す．

```
/*バッファオーバーフローによる脆弱性を持つプログラムのソースコード*/
/*q8-2.c*/
/*プログラム実行時にパスワードを第一引数に与え，パスワード(文字列“clalis”)と一致しているかどうかを判定する*/

#include<stdio.h>
#include<string.h>

/*引数として受け取った文字列が”clalis”と一致していれば 1 を返し，一致していない場合は 0 を返す*/
int valid_passwd(char *pswd){
        char passwd_buffer[16];
        int auth_flag = 0;

        strcpy(passwd_buffer,pswd);

        if( strcmp(passwd_buffer,"clalis") == 0 ){
                auth_flag = 1;
        }

        return auth_flag;
}

int main(int argc,char *argv[]){
        if(argc < 2) return 1;

        if( valid_passwd(argv[1]) ){
                printf("correct password¥n");
        }
        else{
                printf("wrong password¥n");
        }

        return 0;
}
```

```
プログラムの実行例
$gcc q8-2.c
$./a.out aaaaa
wrong password
$./a.out clalis
correct password
$./a.out AAAAAAAAAAAAAAAAAAAA
correct password
```

実行例を見ると引数が”AAAAAAAAAAAAAAAAAAAA”の場合に意図しない出力結果となっている．これは valid_passwd()関数内の passwd_buffer がバッファオーバーフローを起こし，auth_flag の初期値が 0 ではなくなっているためである．これは，返り値の値が authflag の値であり，その値は 0 ではないため main()関数内の if 文分岐で予期しない動きをするためである．

【回答ここまで】

□選択問題 9 （左側の□について、回答した問題は■にしてください）

以下のコードは、与えられたテキスト内から URL らしき文字列を探して、それらを<a>要素でリンクにした HTML を生成する JavaScript の関数であるとします。攻撃者が引数 text の中身を自由に制御可能な場合、このコードにはどのような問題点があるか、またこのコードを修正するとすればどのようにすればよいか、自分なりに考察して書いてください。

```
function makeUrlLinks( text ){
 var html = text.replace( /[¥w]+:¥/¥/[¥w¥.¥-]+¥/[^¥r¥n ¥t<>"']*/g, function( url ){
   return "<a href=" + url + ">" + url + "</a>";
   } );
 document.getElementById( "output" ).innerHTML = html;
}
```

【以下に回答してください（行は適宜追加してください）】

【回答ここまで】

□選択問題 10 （左側の□について、回答した問題は■にしてください）

アンチデバッグ、難読化といった単語をキーワードとする技術について、あなたが知っていること、調べたことを具体的に記述してください。基本的に PC のソフトウェアにおける技術を想定していますが、他端末、またはハードウェアに関する内容でもかまいません。

【以下に回答してください（行は適宜追加してください）】

【回答ここまで】

■選択問題 11 （左側の□について、回答した問題は■にしてください）

下記バイナリを解析し、判明した情報を自由に記述してください

```
D4 C3 B2 A1 02 00 04 00 00 00 00 00 00 00 00 00
00 00 04 00 01 00 00 00 88 EB 40 54 A2 BE 09 00
52 00 00 00 52 00 00 00 22 22 22 22 22 22 11 11
11 11 11 11 08 00 45 00 00 44 1A BD 40 00 80 06
3A 24 C0 A8 92 01 C0 A8 92 80 10 26 01 BB 86 14
7E 80 08 B3 C8 21 50 18 00 FC 0D 0E 00 00 18 03
03 00 17 01 0E FB 06 F6 CD A3 69 DC CA 0B 99 FF
1D 26 09 E1 52 8F 71 77 45 FA
```

【以下に回答してください（行は適宜追加してください）】

アドレス 01～04 が pcap フォーマットのマジックナンバーであることから，このバイナリは何らかのパケットデータを示していることになる．このパケットを受信者が WAN からデフォルトゲートウェイを通して受信した時に解析した状態であると仮定し，wireshark を用いて解析することで以下のことが判明した．

```
D4 C3 B2 A1 02 00 04 00 00 00 00 00 00 00 00 00
00 00 04 00 01 00 00 00 88 EB 40 54 A2 BE 09 00
52 00 00 00 52 00 00 00                         <-ここまで pcap ヘッダー　ここから ethernet ヘッダー
22 22 22 22 22 22                               <-パケットを解析した計算機の MAC アドレス
11 11 11 11 11 11                               <-デフォルトゲートウェイの MAC アドレス
08 00                                           <-上位層のプロトコルの種類(0x0800=IP)
45 00 00 44 1A BD 40 00 80 3A 24                <-ここから IP ヘッダー　バージョンなどの情報
C0 A8 92 01                                     <-パケットを解析した計算機のローカル IP アドレス
C0 A8 92 80                                     <-解析した計算機のデフォルトゲートウェイの IP アドレス
10 26                                           <-ここから TCP ヘッダー　データを送信したアプリケーションのポート番号
01 BB                                           <-データを受け取るアプリケーションのポート番号
86 14 7E 80                                     <-シーケンス番号
08 B3 C8 21                                     <-応答確認番号
50 18                                           <-フラグビット　今回の場合 ACK と PSH が有効
00 FC                                           <-ウィンドウサイズ
0D 0E                                           <-チェックサム
00 00                                           <-緊急ポインタ
18 03 03 00 17                                  <-ここから secure socket layer
01 0E FB 06 F6 CD A3 69 DC CA 0B 99 FF 1D 26 09 <-プロトコルの種類（TLSV1.2）の情報やパケット長の情報など
E1 52 8F 71 77 45 FA                            <-ハートビートメッセージを要求
```

(1) パケットを解析した計算機の MAC アドレスが 22:22:22:22:22:22 であり，その計算機のデフォルトゲートウェイの MAC アドレスが 11:11:11:11:11:11 である．

(2) インターネットプロトコルは TLSV1.2 である．TLS(Transport Layer Security)は，インターネットなどのコンピューターネットワークにおいてセキュリティを要求される通信を行うためのプロトコルである．主な機能として、通信相手の認証、通信内容の暗号化、改竄の検出を提供する。

(3) パケットを解析した計算機のローカルの IP アドレスは 192.168.146.128 であり，その計算機のデフォルトゲートウェイのローカルの IP アドレスは 192.168.146.1 である．

(4) 送信側のプロセスのポート番号は 4134，受信側のプロセスのポート番号は 443 である．

(5) パケットのウィンドウサイズは 252 である．

(6) ACK フラグがオンであり，TCP ヘッダ中に有効な ACK 番号が含まれている．

(7) PSH フラグがオンであり，受信データを速やかに上位アプリケーションに引き渡すよう要求されている．

(8) このパケットはハートビート信号を要求している．

【回答ここまで】

□選択問題 12 （左側の□について、回答した問題は■にしてください）

CentOS 6.5 リリース時点のパッケージを使用して構築された CentOS 6 サーバがあります。ユーザ test のログインシェルには、特定のディレクトリ内のファイルを scp を用いてダウンロードできるようにすることを意図した、以下に示すログインシェルが使われています。

このサーバのホスト名を shell.scamp2015.com とし、/etc/ssh/sshd_config の内容がデフォルト設定のままであると仮定して、このログインシェルの脆弱性と、このログインシェルから起動されるプログラムの脆弱性の両方を突いて、書き込み可能なディレクトリ（好きな場所を仮定してよい）の中に任意のファイルをアップロードする手順を、コマンドラインを交えながら解説してください。

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <wordexp.h>

int main(int argc, char *argv[])
{
    int i;
    int err = EINVAL;
    wordexp_t p;
    char *w;
    static char *args[1024] = { };
    if (argc != 3 || strcmp(argv[1], "-c")) {
        fprintf(stderr, "You are not permitted to login.¥n");
        return 1;
    }
    w = argv[2];
    if (strncmp(w, "scp -f ", 7) ||
        strspn(w + 7, "abcdefghijklmnopqrstuvwxyz0123456789./+-_$[{()}]=`?*'") != strlen(w + 7) ||
        wordexp(w, &p, WRDE_NOCMD) || p.we_wordc < 3 || p.we_wordc > 1023)
        goto out;
    for (i = 0; i < p.we_wordc; i++) {
        w = p.we_wordv[i];
        if (strncmp(w, "/home/", 6) && strncmp(w, "/var/ftp/", 9) &&
            strncmp(w, "/var/www/html/public/", 21) && i >= 2)
            goto out;
        args[i] = w;
    }
    execv("/usr/bin/scp", args);
    err = errno;
out:
    fprintf(stderr, "%s : %s¥n", argv[2], strerror(err));
    return 1;
}
```

【以下に回答してください(行は適宜追加してください)】

【回答ここまで】

□選択問題 13 (左側の□について、回答した問題は■にしてください)

これまでに起こったこと(データ)から、これまでにまだ起こっていないことを事前に予想する場合、どのような点に注意し、どのようなことを考慮すべきか熱烈にアピールしてください。

【以下に回答してください(行は適宜追加してください)】

【回答ここまで】

□選択問題 14 (左側の□について、回答した問題は■にしてください)

以下は WebView を使用した Android アプリのコードです。このコードには、不正なアプリや WebView 上で開かれた悪意のあるページによる攻撃を受け、情報流出などの被害をもたらす恐れのある箇所が存在します。それはどこか、またどのような攻撃を受けた場合にどのような被害が生じるか、可能な限り述べてください。なお、コメント欄に付記した CVE の番号は、類似のコードに起因する過去の脆弱性事例です。これらの脆弱性の原因を調べることが回答の手がかりとなります。

```
public class MainActivity extends Activity {
 private static final String ACTION_LOAD = "jp.example.app.action.LOAD";
 private static final String ALLOW_ORIGIN = "http://www.ipa.go.jp";
 private WebView webview;
 private BroadcastReceiver receiver;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   webview = (WebView) findViewById(R.id.webView);
   webview.getSettings().setJavaScriptEnabled(true);
   webview.getSettings().setAllowUniversalAccessFromFileURLs(true); // CVE-2012-4009
   webview.addJavascriptInterface(this, "android"); // CVE-2012-6636
   webview.setWebViewClient(new WebViewClient() {
     @Override
```

```
    public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {
      handler.proceed(); // CVE-2014-5991
    }
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) { // CVE-2014-1883
      Pattern p = Pattern.compile(ALLOW_ORIGIN); // CVE-2012-6637
      Matcher m = p.matcher(url);
      return !(m.find());
    }
  });
  webview.loadUrl(ALLOW_ORIGIN);
  IntentFilter filter = new IntentFilter(ACTION_LOAD);
  receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      webview.loadUrl(intent.getStringExtra("url")); // CVE-2014-3500
    }
  };
  registerReceiver(receiver, filter);
 }

 @Override
 protected void onDestroy() {
  unregisterReceiver(receiver);
 }
}
```

【以下に回答してください(行は適宜追加してください)】

【回答ここまで】