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．

1. 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．

1. 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

$

1. 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． |

1. 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．

1. 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．
2. 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□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．

/\*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;

}

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

#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)

buf[5]の領域

カナリア値

32bit

5

0x8049750

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;

}

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．

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

【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】

※応募用紙にご記入いただいた、事務局が取得した個人情報を含む一切の内容につきましては、独立行政法人情報処理推進機構（IPA）の個人情報保護方針に基づき、セキュリティ・キャンプ実施協議会と共に適正に取扱いを行います。取得した情報は、セキュリティ・キャンプ関連の連絡・案内、資料送付、セキュリティ・キャンプ実施協議会からの案内など、「セキュリティ・キャンプ」の事業に関連する用途においてのみ使用し、それ以外の目的には一切使用しません。詳しくは、ホームページをご覧ください。http://www.ipa.go.jp/about/privacypolicy/index.html