セキュアコーディングノススメ

2008年1月24日(木) @
JANOG21 Kumamoto

JPCERT コーディネーションセンター
情報セキュリティアナリスト
戸田　洋三
久保　正樹

JPCERT CC®

セキュアコーディング・クイズ！
の回答と解説

JPCERT CC®

皆さん事前資料は見てきましたよね？！

脆弱性のトレンド、いまなぜセキュアコーディングか、については、事前資料のスライド11, 20, 21あたりを見といてね！

クイズの答えを解説します！

全問正解の人にはすてきな景品が！

問題1　sizeof() の不適切な使用はどこ？

```
void func(char *s) {
  size_t size = sizeof(s) / sizeof(s[0]);
}

int main(void) {
  char str[] = "Bring on the dancing horses";
  size_t size = sizeof(str) / sizeof(str[0]);
  func(str);
}
```

回答1　sizeof() の不適切な使用はどこ？

```
void func(char  」」*s) {
  size_t size = sizeof(s) / sizeof(s[0]);
}

int main(void) {
  char str[] = "Bring on the dancing horses";
  size_t size = sizeof(str) / sizeof(str[0]);
  func(str);
}
```

サイズは 4

サイズは 1

サイズは 28

サイズは 1

問題2 オフバイワンエラーが発生するのはどこ？

```
int main(void) {
 char source[10];
 strcpy(source, "0123456789");
 char *dest = malloc(strlen(source));
 for (int i=1; i <= 11; i++) {
  dest[i] = source[i];
 }
 dest[i] = '¥0';
 printf("dest = %s", dest);
}
```

回答2 オフバイワンエラーが発生するのはどこ？

```
int main(void) {
 char source[10];
 strcpy(source, "0123456789");
 char *dest = malloc(strlen(source));
 for (int i=1; i <= 11; i++) {
  dest[i] = source[i];
 }
 dest[i] = '¥0';
 printf("dest = %s", dest);
}
```

サイズが10の配列に対して、10文字+NULL終端の11文字をコピーしようとしている！

カウンタが1からスタートするのがおかしい！

i が 11のときに境界外参照が発生！

更に、境界外書き込みも行ってしまう！

問題3 境界外へ書き込みが行われる可能性があるのはどこ？

Kerberos 5 Version 1.0.6 に実際にあった脆弱性

「

```
if (auth_sys == KRB5_RECVAUTH_V4) {
  strcat(cmdbuf, “/v4rcp”);
} else {
  strcat(cmdbuf, “/rcp”);
}
if (stat((char *)cmdbuf + offst, &s) >= 0)
  strcat(cmdbuf, cp);
else
  strcpy(cmdbuf, copy);
```

回答3 境界外へ書き込みが行われる可能性があるのはどこ？

```
if (auth_sys == KRB5_RECVAUTH_V4) {
  strcat(cmdbuf, “/v4rcp”);
} else {
  strcat(cmdbuf, “/rcp”);
}
if (stat((char *)cmdbuf + offst, &s) >= 0)
  strcat(cmdbuf, cp);
else
  strcpy(cmdbuf, copy);
```

cp のサイズによっては cmdbuf がオーバーフローする

copy のサイズによっては cmdbuf がオーバーフローする

回答3つづき 実際に修正された Kerberos 5 のコード

```
cmdbuf[sizeof(cmdbuf) - 1] = '¥0'
if (auth_sys == KRB5_RECVAUTH_V4) {
  strncat(cmdbuf, "/v4rcp", sizeof(cmdbuf) -1 -
strlen(cmdbuf));
} else {
  strncat(cmdbuf, "/rcp", sizeof(cmdbuf) -1 -
strlen(cmdbuf));
}
if (stat((char *)cmdbuf + offst, &s) >= 0)
  strncat(cmdbuf, cp, sizeof(cmdbuf) -1 - strlen(cmdbuf));
else
  strncpy(cmdbuf, copy, sizeof(cmduf) -1 -
strlen(cmdbuf));
```

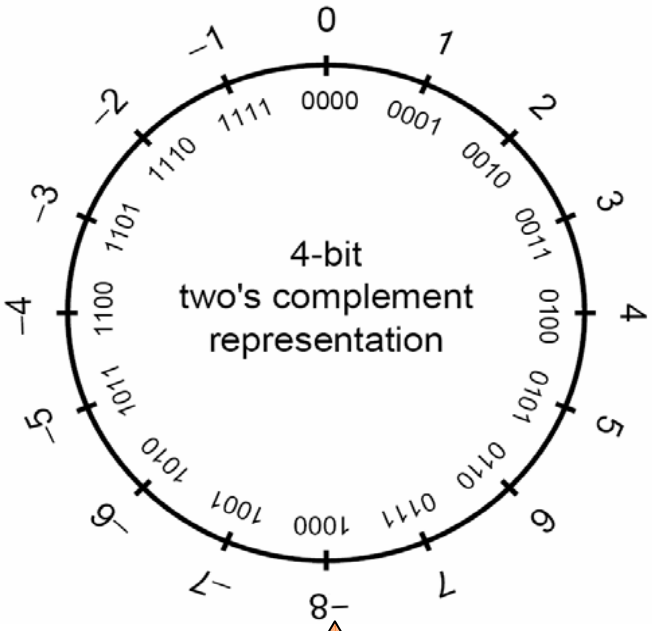
問題4 puts() が実行されるのはどの行？

```
signed char x, y;
x = -128;
y = -x;

if (x == y) puts("1");
if ((x - y) == 0) puts("2");
if ((x + y) == 2 * x) puts("3");
if (((char)(-x) + x) != 0) puts("4");
if (x != -y) puts("5");
```

回答4 puts() が実行されるのはどの行？

```
signed char x, y;
x = -128;
y = -x;

if (x == y) puts("1");
if ((x - y) == 0) puts("2");
if ((x + y) == 2 * x) puts("3");
if (((char)(-x) + x) != 0) puts("4");
if (x != -y) puts("5");
```

式 -x は整数拡張され128 (0...0100) となるが、signed char の幅に代入されると -128 として解釈される！



x = −128

演算 != を行うまえにオペランド x, y は整数拡張されるため、x = −128, y = 128 として評価される。

char にキャストしているので、演算時に整数拡張(integer promotion)が発生しない！
つまり、-x は x と等しくなる。

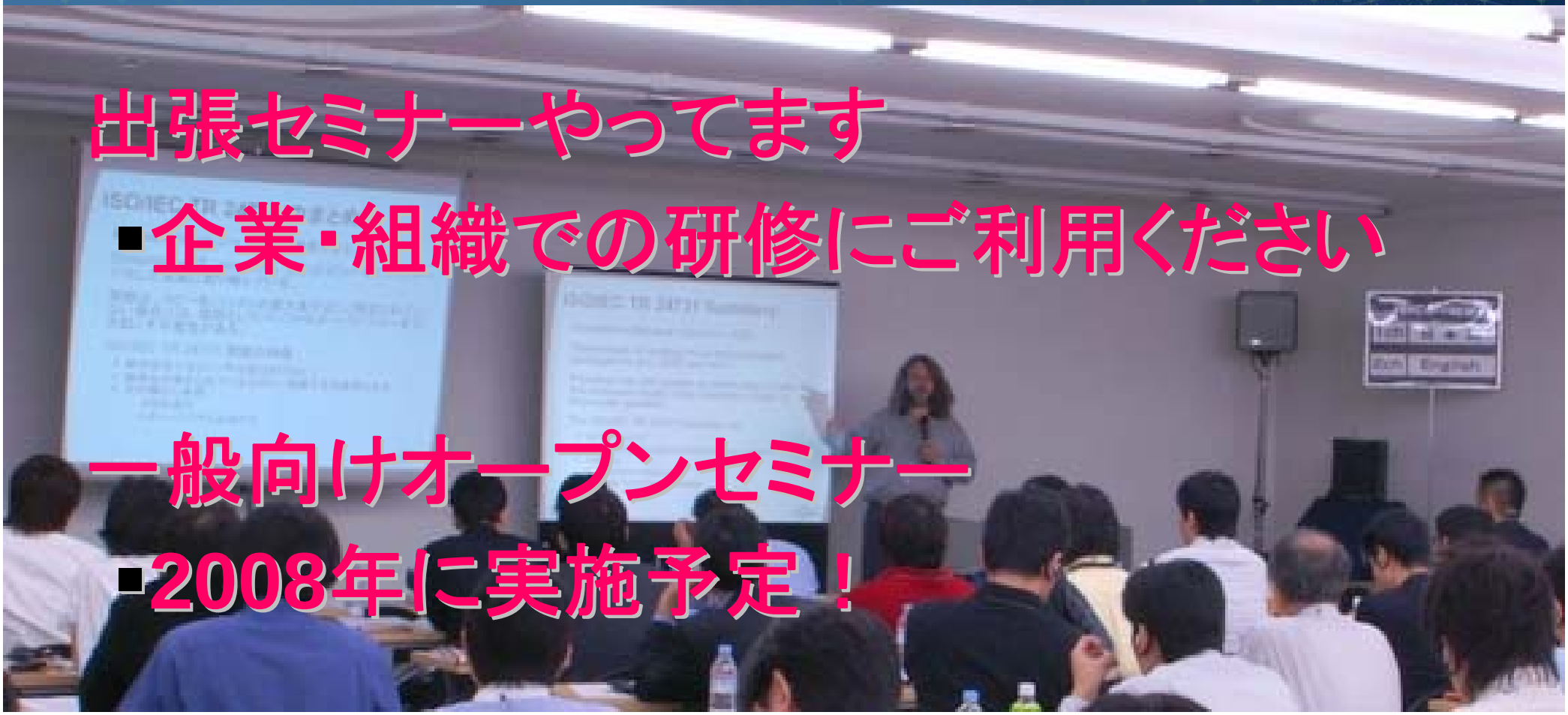
まとめ

- 信頼できない入力ソースから入ってくる値は、以下の点をチェックする
  - 値のとりうる上限、下限は特定できるか
  - 可能な場合、入力インターフェイスから強制的にチェックを行う

- 整数の脆弱性を回避するには、

Cの仕様を深〜く理解しておかないとダメよ。

セキュアコーディングセミナー、トレーニングのご案内



出張セミナーやってます

- 企業・組織での研修にご利用ください

一般向けオープンセミナー

- 2008年に実施予定！

※写真は昨年 Robert C Seacord 氏を迎えたセミナの模様

詳しくはJPCERT/CCスタッフへお問合せください