

**Tugas Akhir**  
**Mata Kuliah Keamanan Sistem Informasi**

***Secure Coding* pada Bahasa Pemograman C/C++  
(String dan Integer)**

Oleh :

**Khairul Hamdi**  
**23206331**



**Program Studi Teknik Elektro**  
**Sekolah Tinggi Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2007**

## DAFTAR ISI

<b>Abstract</b> .....	1
<b>Bab I Pendahuluan</b>	
A. Latar belakang.....	2
B. Tujuan.....	2
C. Batasan Masalah.....	3
<b>Bab II Kerawanan pada String</b>	
A. Pengertian.....	4
B. Pengkopian string yang tidak terbatas.....	5
C. Kesalahan pengakhiran null.....	8
D. Pemenggalan string.....	8
E. Penanggulangan.....	9
<b>Bab III Kerawanan pada Integer</b>	
A. Pengertian.....	10
B. <i>Overflow</i> .....	11
C. Pemenggalan ( <i>truncation</i> ).....	11
D. Kesalahan tanda.....	12
E. Penanggulangan.....	12
<b>Bab IV Penutup</b>	
A. Kesimpulan .....	14
B. Saran.....	14
<b>Daftar Pustaka</b> .....	15

## ***Abstract***

*C and C++ are popular language programming that used to develop many application because it's flexibility and performance. But security factor has increasingly become an issue. In C and C++, string representated by null-terminated array of character. Weakness in string representation, string management and string manipulation have caused a broad range of software vulnerabilities and exploits. These problem cause unbounded string copies, null termination erros and string truncation error. To avoid these, programer can use dinamic memory allocation to allocate buffer and resize buffer when needed. An inherents problem in computing is that digital representation of integer always limited by range of values they can represent. As result, problem can rise such as integer overflow, truncation and sign error. Preventif action is check range of integer that be used and use typographic convention.*

# BAB I

## PENDAHULUAN

### A. Latar Belakang

Bahasa pemrograman C dan C++ merupakan bahasa pemrograman tingkat tinggi yang banyak digunakan untuk membuat berbagai perangkat lunak. Keandalan bahasa ini telah dibuktikan dengan telah banyaknya perangkat lunak yang dihasilkan seperti Apache dan PHP. Sistem operasi Unix juga dibangun menggunakan bahasa C. Bahasa pemrograman C merupakan bahasa pemrograman terstruktur yang membagi program dalam sejumlah blok. Program yang ditulis dengan bahasa C dapat dipindahkan dari satu jenis mesin ke mesin yang lain karena adanya standarisasi ANSI yang menjadi acuan oleh para pembuat kompilator C.

Sebagai turunan dari C, bahasa C++ sebagai bahasa yang berorientasi obyek yang mempunyai fitur yang lebih kompleks dibanding pendahulunya. Berbagai fitur pada C sudah terangkum pada C++ ditambah fitur-fitur baru yang mendukung pemrograman berorientasi obyek. Namun C++ bukanlah bahasa yang murni berorientasi obyek, tetapi merupakan bahasa hibrid antara bahasa pemrograman terstruktur dan bahasa pemrograman berorientasi obyek.

Walaupun bahasa C/C++ mempunyai banyak kelebihan, namun dalam bahasa pemrograman ini masih banyak terdapat lubang kerawanan yang bisa berakibat fatal bagi perangkat lunak yang dibangun. Salah satu kerawanan yang dapat terjadi adalah pada saat penggunaan tipe data string dan integer. Integer dan string merupakan pustaka-pustaka yang dibangun oleh hampir semua bahasa pemrograman tingkat tinggi. Pada beberapa bahasa pemrograman, manajemen string sudah terkelola dengan baik. Namun pada C/C++ masih terdapat beberapa kerawanan yang memerlukan penanganan yang hati-hati. Berbeda dengan tipe string pada bahasa pemrograman lain, pada bahasa C/C++ string merupakan tipe data array yang diakhiri dengan karakter *null*. Kerawanan yang terjadi pada aplikasi yang dibangun dengan C/C++ salah satunya disebabkan oleh penanganan yang kurang cermat terhadap terminasi *null* ini. Fungsi-fungsi yang menangani tipe data string pada C/C++ juga berpotensi menimbulkan kerawanan bila tidak ditangani secara hati-hati. Pada pustaka C/C++ terdapat berbagai tipe integer yang mempunyai jangkauan yang berbeda-beda. Tipe data integer juga dapat dibagi menjadi tipe integer *signed* dan *unsigned*. Kerawanan dapat terjadi bila keliru dalam menentukan tipe data integer yang digunakan.

### B. Tujuan

Tujuan dari penulisan makalah adalah :

1. Memahami kerawanan-kerawanan yang mungkin terjadi saat penggunaan tipe string pada bahasa C/C++.
2. Memahami kerawanan-kerawanan yang mungkin terjadi saat penggunaan tipe integer pada bahasa C/C++.
3. Mengetahui pencegahan kerawanan pemakaian tipe data string dan integer pada bahasa C/C++.

### **C. Batasan Masalah**

Pada tulisan ini, masalah dibatasi sebagai berikut :

1. Obyek yang diamati adalah tipe data string dan integer pada bahasa pemograman C dan C++.
2. Pembahasan difokuskan pada faktor keamanan dan menganalisis kerawanan yang mungkin terjadi.

## BAB II KERAWANAN PADA STRING

### A. Pengertian

String adalah tipe data pada bahasa pemrograman C/C++ yang berfungsi dalam pertukaran data antara pengguna dan sistem perangkat lunak. String terdiri dari argumen pada perintah baris, variabel pada suatu lingkungan dan masukan konsol. Format string adalah gabungan dari karakter yang digandakan pada bagian keluaran dan dapat juga menjadi penentu konversi yang berfungsi sebagai karakter pengganti untuk argumen selanjutnya.

String pada bahasa pemrograman C/C++ terdiri dari karakter berurutan yang diakhiri dengan karakter null. Konstanta string disimpan dalam memori dalam bentuk array dimana setiap karakter menempati memori sebesar satu byte. Deklarasi tipe data string pada bahasa pemrograman C/C++ sama seperti mendefinisikan array dengan tipe karakter. String pada bahasa pemrograman C/C++ dinyatakan sebagai berikut :

h	a	l	l	o	\0
---	---	---	---	---	----

*Gambar 1. Penyimpanan string dalam memori*

Untuk menyatakan string pada bahasa C/C++, pointer akan menunjuk karakter awal kemudian akan menunjuk karakter selanjutnya sampai menunjuk null. Panjang string adalah jumlah karakter yang terdapat sebelum karakter null (\0).

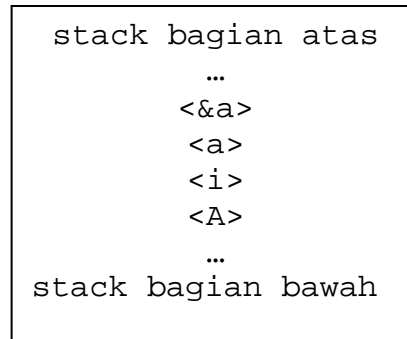
String merupakan konsep yang fundamental pada perangkat lunak, tapi pada bahasa pemrograman C/C++ elemen ini tidak terdapat secara *built in*. Kerawanan yang terjadi pada perangkat lunak dan kesalahan yang terjadi umumnya disebabkan oleh representasi, manajemen string dan manipulasi string.

Ada berbagai aplikasi yang mempunyai kerawanan pemakaian tipe data string. Beberapa perangkat lunak yang mempunyai kerawanan sehubungan dengan pemakaian tipe string terlihat pada Tabel 1.

Aplikasi	Ditemukan oleh	Dampak	lama penggunaan
wu-ftpd 2.	security.is	<i>remote root</i>	6 tahun
Linux rpc.statd	security.is	<i>remote root</i>	4 tahun
IRIX telnetd	LSD	<i>remote root</i>	8 tahun
Qualcomm Popper 2.53	security.is	<i>remote user</i>	3 tahun
Apache + PHP3	security.is	<i>remote user</i>	2 tahun
NLS / locale	CORE SDI	<i>local root</i>	-
Screen	Jouko Pynnoen	<i>local root</i>	5 tahun
BSD chpass	TESO	<i>local root</i>	-
OpenBSD fstat	ktwo	<i>local root</i>	-

*Tabel 1 . Aplikasi perangkat lunak yang mempunyai kerawanan dalam pemakaian tipe data string pada tahun 2000*

Pada tabel diatas bahwa pada beberapa aplikasi, terdapat kerawanan pemakaian tipe data string. Aplikasi yang telah umum dipakai diseluruh dunia seperti *web server* Apache dan PHP ternyata mempunyai kelemahan dalam pemakaian tipe data string sebagaimana yang ditemukan oleh *security.is*. String dalam memori disimpan dalam bentuk *stack* seperti terlihat pada gambar 2.



Gambar 2 . Memori stack untuk menyimpan tipe data string.

Gambar 2 merupakan sebuah *stack* dengan keterangan sebagai berikut :

- A = alamat format string
- i = nilai variabel i
- a = nilai variabel a
- &a = alamat variabel i

Standar pada C++ dinyatakan dengan standar kelas templet `std::basic_string` dan diikuti dengan `std::string`. Kelas `basic_string` mempunyai tingkat kerawanan yang lebih rendah dibanding string pada bahasa C. String pada bahasa C merupakan tipe data umum pada bahasa C++.

Kesalahan yang umum terjadi pada tipe string pada bahasa C++ adalah pengkopian string yang tidak terbatas, kesalahan pengakhiran null dan pemenggalan data atau *truncating*.

## B. Pengkopian string yang tidak terbatas

Kesalahan ini terjadi ketika data dikopikan dari sumber yang besar atau tidak terbatas kedalam array yang mempunyai panjang yang tetap. Proses ini menyebabkan data dari sumber akan dikopikan semuanya. Sebagai contoh terlihat pada program dibawah ini.

```
void main(void)
{
    char password[80];
    puts("Masukan 8 karakter password :");
    gets(password);
}
```

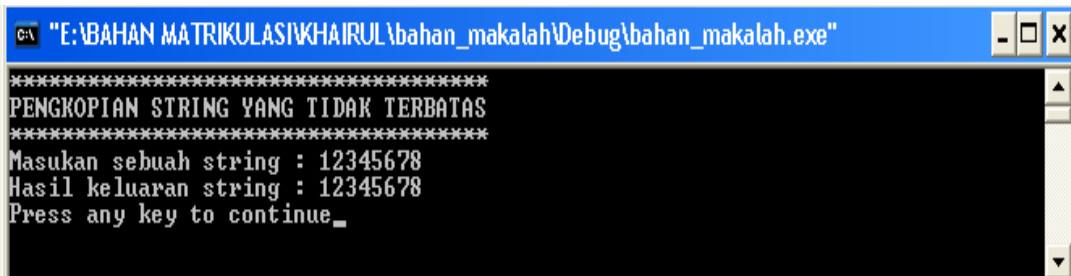
Pada program diatas, deklarasi `password[80]` menyatakan sebuah string dengan isi maksimal 80 karakter. Fungsi `gets(password)` akan mengkopi seluruh isi yang ada pada string sebanyak 80 karakter walaupun jumlah karakter masukan kurang dari 80. String

yang dikopikan dari sumber yang besar atau tidak terbatas ke array yang mempunyai panjang tetap akan menyebabkan terjadinya pengkopian yang tidak terbatas.

Pada saat pengkopian string pada C++, bisa terjadi *error* bila jumlah elemen yang dimasukkan pada sebuah array lebih besar dari jumlah komponen maksimal yang bisa dimasukkan kedalam array tersebut. Setelah sebuah string dimasukkan kedalam sebuah array yang komponennya melebihi kapasitas array tersebut maka bila dipanggil kembali akan terjadi pengkopian yang tidak terbatas seperti pada cuplikan program berikut ini.

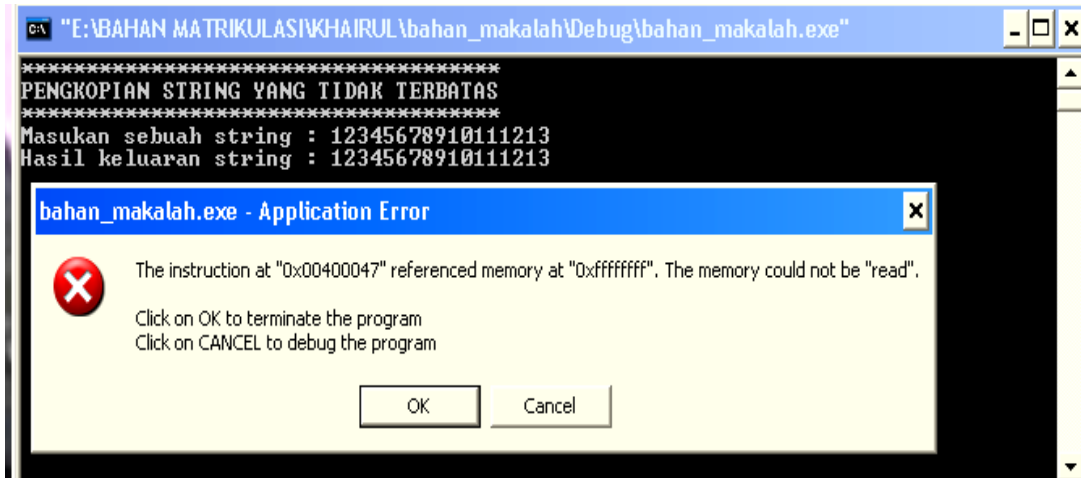
```
#include<iostream>
int main()
{
    char STRING[12];
    cout<<"*****"<<endl
        <<"PENKOPIAN STRING YANG TIDAK TERBATAS"<<endl
        <<"*****"<<endl;
    cin>>STRING;
    cout<<" Keluaran : "<<STRING<<endl;
}
```

Bila jumlah karakter yang dimasukkan kecil atau sama dengan 12, maka tidak akan terjadi *error* karena string dapat dikopi seluruhnya. Namun bila jumlah karakter yang dimasukkan lebih besar dari 12 karakter maka akan terjadi *error*. *Error* ini terjadi karena argumen `STRING[12]` membatasi karakter yang disimpan dalam memori sebanyak 12 karakter.



Gambar 3. Hasil program pengopian yang tidak terbatas

Bila karakter yang dimasukkan lebih besar dari jumlah array maka akan terjadi *error* seperti pada gambar 4 .



Gambar 4. Kesalahan yang terjadi pada kasus pengkopian yang tidak terbatas

Sebagai solusi dalam permasalahan tersebut, operasi masukan bisa dibatasi dengan dengan memakai librari `ios_base::width` yang diset untuk membatasi nilai maksimal suatu masukan. Cuplikan programnya dapat dilihat dibawah ini.

```
#include<iostream>
int main()
{
    char STRING[12];
    cin.width(12);
    cout<<"*****"<<endl
        <<"PENGKOPIAN STRING YANG TIDAK TERBATAS"<<endl
        <<"*****"<<endl;
    cin>>STRING;
    cout<<" Keluaran : "<<STRING<<endl;
}
```

Dengan mendeklarasikan fungsi `cin.width(12)` akan membatasi jumlah karakter masukan hanya boleh 12 karakter.

Kerawanan pada string juga dapat terjadi pada saat proses konkatenasi. Kerawanan terjadi karena fungsi standar string tidak mengetahui ukuran buffer. Bila ukuran buffer penyangga tidak diketahui, menyebabkan pointer akan menunjuk kedalam memori yang tidak menyimpan karakter, hal ini bisa menjadi lubang keamanan pada program atau perangkat lunak yang dikembangkan menggunakan bahasa C/C++. Sebagai contoh dapat dilihat pada cuplikan program dibawah ini :

```
#include<iostream>
#include<string.h>
{
    char nama[2048];
    strcpy(nama,argv[1]);
    strcat(nama,"=");
    strcat(nama,argv[2]);
}
```

Pada program diatas, akan terjadi *error* waktu dijalankan karena ukuran buffer tidak diketahui oleh pointer.

### C. Kesalahan Pengakhiran null

Tipe data string pada C/C++ merupakan tipe data array yang diakhiri dengan karakter *null*. Bila suatu string tidak diakhiri dengan karakter *null* bisa menyebabkan *error*. Hal ini bisa terjadi pada bahasa C/C++ dimana pada waktu pembacaan suatu string tidak diakhiri dengan karakter *null*. Sebagai contoh dapat dilihat pada program berikut ini.

```
int main(int argc, char* argv[])
{
    char a[16];
    char b[16];
    char c[32];
    strcpy(a, "0123456789abcdef", sizeof(a));
    strcpy(b, "0123456789abcdef", sizeof(b));
    strcpy(c, a, sizeof(c));
}
```

Pada program diatas, string `a[16]` dan `b[16]` tidak mengalami pengakhiran null yang wajar.

### D. Pemenggalan String

Pemenggalan string terjadi karena adanya *string overflow*. *String overflow* terjadi ketika string dituliskan kedalam buffer yang mempunyai panjang yang tetap. Bila jumlah karakter yang dimasukan kedalam buffer lebih banyak dari tempat yang tersedia, maka karakter yang berlebih akan terpenggal. Kebanyakan bahasa pemograman tingkat tinggi bisa mengatasi masalah ini dengan menyeting kembali ukuran array bila kasus *overflow* terdeteksi. Sehingga kesalahan yang terjadi karena *overflow* bisa dicegah, seperti pada bahasa Perl dan Ada95. Namun fasilitas ini tidak tersedia pada C/C++ sehingga harus digunakan fungsi-fungsi khusus.

Pada *library* C/C++ banyak terdapat fungsi-fungsi yang tidak mengecek batas array dari nilai yang dimasukan. Tidak adanya pengecekan terhadap batas nilai bisa menyebabkan terpenggalnya nilai ada pada array. Fungsi-fungsi yang dimaksud seperti `strcpy()`, `strcat()`, `sprintf()`. Fungsi-fungsi tersebut dapat digunakan bila karakter yang disimpan dalam string jumlahnya tidak melebihi batas yang telah ditentukan. Dibawah ini merupakan contoh program terjadinya pemenggalan karakter yang disebabkan oleh *string overflow*.

```
int main(int argc, char *argv[]){
    int i=0;
    char buff[128];
    char *arg1=argv[1];
    while (arg[i] !='\0'){
        buff[i]=arg[i];
    }
    buff[i]='\0';
    printf("buff=%s\n",buff);
}
```

Pada cuplikan program diatas, pointer akan memeriksa nilai-nilai yang ada dalam memori, bila ditemukan karakter null, pointer akan berhenti. Namun nilai yang ditampilkan maksimal 128 karakter karena argumen `char buff[128]`, ukuran buffer telah dibatasi 128 karakter. Bila nilai yang diberikan lebih besar dari 128 karakter, maka karakter yang berlebih akan dipenggal.

Untuk mengatasi masalah tersebut, pemakaian fungsi-fungsi diatas dapat diganti dengan `strncpy()`, `strncat()`, `snprintf()` dan `fgets()`. Pemakaian fungsi ini tidak membatasi jumlah karakter yang digunakan sehingga tidak terjadi pemenggalan karakter. Terjadinya pemenggalan karakter bisa menyebabkan kerusakan data. Pada beberapa kasus juga menyebabkan kerawanan pada aplikasi yang dibangun.

## **E. Penanggulangan**

Untuk meningkatkan keamanan pada aplikasi yang dibangun memakai bahasa C/C++ dapat dilakukan dengan membuat string yang selalu diakhiri dengan *null*, membuat string yang tidak bisa mengalami pemotongan dan membuat program dengan pola yang seragam untuk parameter fungsi dan tipe kembalian.

Pengaturan string dapat dilakukan secara dinamis, teknik yang dapat digunakan adalah mengalokasikan buffer dan menset kembali ukuran string dengan penambahan memori jika diperlukan. Operasi manajemen string akan menjamin bahwa operasi string tidak akan dihasilkan pada saat buffer overflow, data tidak mengalami pembuangan, string tidak mengalami pengakhiran null secara internal. Namun metode ini mempunyai beberapa kelemahan diantaranya bisa menghabiskan memori dan bila ada serangan *denial of service*, kinerjanya akan menurun.

## BAB II KERAWANAN PADA INTEGER

### A. Pengertian

Tipe integer merupakan fitur utama yang terdapat pada bahasa pemrograman C/C++. Namun tipe integer ini mengalami tingkat kerawanan yang tinggi karena kebanyakan perangkat lunak yang dibangun menggunakan bahasa C/C++ tidak memperhatikan jangkauan bilangan integer yang digunakan. Banyak kerusakan keamanan yang terjadi ada kaitannya dengan keberadaan tipe data integer. Kerawanan pada perangkat lunak dapat terjadi ketika program mengevaluasi tipe integer kepada nilai yang tidak dapat diprediksi.

Tipe data integer yang tersedia dalam bahasa pemrograman C/C++ adalah tipe data *signed* dan *unsigned*. Bilangan *integer signed* digunakan menampilkan bilangan yang bernilai positif dan negatif. Jangkauan bilangan *integer signed* dari  $-2^{n-1}$  sampai  $2^{n-1}$ ,  $n$  menyatakan bit yang digunakan.

Bilangan *integer unsigned* dari 0 sampai  $2^{n+1}$ ,  $n$  menyatakan jumlah bit yang digunakan untuk menyatakan tipe bilangan *integer unsigned*. Standar tipe-tipe bilangan *integer* yang digunakan dalam bahasa C++ adalah sebagai berikut :

- signed char
- short int
- int
- long int
- long long int

No.	Dari (unsigned)	Ke	Implikasi
1	char	char	pola bit dipertahankan, bit orde tertinggi menjadi bit penanda
2	char	short	perpanjangan bit kosong
3	char	long	perpanjangan bit kosong
4	char	unsigned short	perpanjangan bit kosong
5	char	unsigned long	perpanjangan bit kosong
6	short	char	byte berorde rendah dipertahankan
7	short	short	pola bit dipertahankan, bit orde tinggi menjadi bit penanda
8	short	long	perpanjangan bit kosong
9	short	unsigned char	byte berorde rendah dipertahankan
10	long	char	byte berorde rendah dipertahankan
11	long	short	karakter berorde rendah dipertahankan
12	long	long	pola bit dipertahankan, bit orde tinggi menjadi bit penanda
13	long	unsigned char	byte berorde rendah dipertahankan
14	long	unsigned short	karakter berorde rendah dipertahankan

*Tabel 2. Tipe-tipe konversi pada tipe data integer*

Nilai maksimum dan minimum untuk bilangan tipe *integer* tergantung kepada tipe yang digunakan, penandaan, dan jumlah alokasi bit.

Tipe konversi terjadi secara eksplisit pada bahasa pemrograman C/C++ sebagai implikasi suatu operasi. Operasi konversi bisa menyebabkan kehilangan dan misinterpretasi data. Konversi implisit terjadi sebagai konsekuensi dari kemampuan bahasa C untuk melakukan operasi pada tipe data yang berbeda. Tipe-tipe konversi yang terjadi pada tipe data *char* dan *integer* beserta dampaknya terlihat pada tabel 1. Konversi dari tipe data dengan bit yang lebih besar ke tipe data dengan bit yang lebih kecil akan mengakibatkan kehilangan data seperti konversi tipe data pada nomor 6, 9, 10, 11, 12, 13. Konversi data dari tipe data *integer unsigned* ke *integer signed* dengan jumlah bit yang sama menyebabkan misinterpretasi data seperti yang terjadi pada konversi nomor 7 dan 12. Operasi pada bilangan *integer* bisa menyebabkan nilai yang berbeda karena *overflow*, pemenggalan data (*truncation*) dan kesalahan tanda.

## B. *Overflow*

*Overflow* pada tipe data *integer* terjadi apabila nilai data naik melebihi nilai maksimum atau nilai turun melebihi nilai minimum. *Overflow* bisa terjadi untuk bilangan *integer signed* atau *unsigned*. *Overflow* bilangan *integer signed* terjadi apabila nilai melebihi tanda bit dan *overflow* bilangan *integer unsigned* terjadi apabila data yang representasikan disimpan dalam tipe data yang mempunyai nilai bit yang lebih kecil.

Cuplikan program dibawah ini menggambarkan *overflow* yang terjadi pada tipe bilangan *integer*.

```
int i;
unsigned int j;
i = INT_MAX;
i++;
printf("i = %d\n", i);
j = UINT_MAX;
j++;
printf("j = %u\n", j);
```

Argumen `INT_MAX` menyatakan nilai maksimum tipe data *signed integer* yaitu 2.147.483.648, bila operator diberikan *increment* akan dihasilkan keluaran -2.147.483.648 karena terjadi *overflow*. Argumen `UINT_MAX` menyatakan nilai maksimum tipe data *unsigned integer* yaitu 4.294.967.295, bila diberikan operator *increment* akan terjadi *overflow* dengan nilai keluaran 0.

## C. Pemenggalan (*truncation*)

Pemenggalan terjadi ketika nilai suatu data bertipe *integer* dikonversikan menjadi tipe data yang lebih kecil. Kesalahan pemenggalan juga terjadi bila nilai awal *integer* berada diluar jangkauan tipe data *integer* yang lebih kecil. Pada peristiwa *truncation* bit yang mempunyai orde yang lebih rendah akan dilewatkan sedangkan nilai dengan orde bit yang lebih tinggi akan mengalami pemenggalan. Contoh *truncation* dapat dilihat pada cuplikan program dibawah ini.

```

char hasil, c1, c2;
c1=100;
c2=90;
hasil=c1+c2;

```

Pada cuplikan program diatas `hasil`, `c1` dan `c2` mempunyai tipe data *signed char* dengan jangkauan -127 sd. +127. Nilai variabel `hasil` adalah 190 yang berada diluar jangkauan tipe data. *Truncation* terjadi karena nilai yang dihasilkan ditempatkan pada tipe data yang terlalu kecil.

#### D. Kesalahan tanda

Bila sebuah nilai *unsigned* dikonversikan kenilai yang *signed* dengan panjang yang sama, maka pola bit akan tetap dipertahankan dan bit dengan orde yang lebih tinggi menjadi bit tanda. Nilai yang berada diatas nilai maksimum pada bilangan *integer signed* akan diubah menjadi bilangan negatif.

```

int i = -3;
unsigned short u;
u = i;
printf("u = %hu\n", u);

```

Pada cuplikan program diatas terjadi proses sebaliknya, nilai *integer signed* diubah menjadi *unsigned integer*. Susunan bit tetap dipertahankan, semua bit pada tipe *signed integer* termasuk bit yang menyatakan tanda diubah semuanya menjadi bit yang menyatakan bilangan *integer*. Pada cuplikan program diatas, nilai keluaran menjadi 65533.

#### E. Penanggulangan

Untuk memecahkan persoalan diatas, ada beberapa teknik yang dapat dipakai, diantaranya :

##### 1. Pengecekan terhadap range yang digunakan.

Pengecekan terhadap *range* yang digunakan bisa menghindari kerawanan yang mungkin terjadi. Masukan eksternal harus dievaluasi untuk menentukan batas atas dan batas bawah, dengan menggunakan batas atas dan batas bawah maka data akan mudah mendeteksi dan memperbaiki kesalahan yang berkaitan dengan melakukan penelusuran terhadap batas atas dan batas bawah tipe data integer yang digunakan. Kemungkinan terjadinya *truncation* pada tipe data integer dapat dilakukan dengan membuat batas atas tipe integer yang lebih tinggi.

##### 2. Membuat konvensi tipografik.

Konvensi tipografik adalah perjanjian penggunaan karakter yang digunakan. Sebagai contoh terdapat perbedaan karakter yang digunakan untuk menyatakan tipe variabel dan konstanta. Untuk variabel biasanya digunakan huruf kecil sedangkan untuk konstanta digunakan huruf kapital seperti contoh dibawah ini.

```
#include "stdafx.h"
#include<iostream.h>
int main (int argc, char*argv[])
{
    int jari_jari;
    double PHI=3.14, keliling;
    cout<<"Masukan jari-jari";cin>>jari_jari;
    keliling=PHI*jari_jari*jari_jari;
    cout<<"Keliling = "<<keliling<<endl;
    return (0);
}
```

Pada cuplikan program diatas lebih mudah untuk menentukan variabel atau konstanta karena perbedaan tanda antara variabel dan konstanta.

## **BAB IV PENUTUP**

### **A. Kesimpulan**

Secara umum kerawanan-kerawanan yang terjadi pada tipe data string pada bahasa pemrograman C/C++ adalah pengkopian yang tidak terbatas, kesalahan pengakhiran null, string overflow dan pembersihan data yang tidak sempurna. Kesalahan ini terjadi karena representasi, manajemen dan manipulasi yang kurang baik. Pada C/C++ terdapat penggunaan fungsi pada string yang menyebabkan kerawanan pada perangkat lunak yang dibangun. Untuk itu C/C++ menyediakan berbagai fungsi khusus untuk menanggulangi kerawanan yang mungkin terjadi. Sebagai pencegahannya, programmer juga harus hati-hati memakai fungsi-fungsi yang ada pada string.

Kesalahan umum yang terjadi pada pemakaian tipe data integer pada C/C++ adalah pemakaian tipe data integer yang kurang memperhatikan jangkauan nilainya. Kesalahan yang mungkin terjadi adalah *overflow*, pemenggalan (*truncation*) dan kesalahan tanda. Pencegahan dapat dilakukan dengan mengecek jangkauan tipe data yang digunakan dan mengikuti konvensi tipografik.

### **B. Saran**

Walaupun bahasa pemrograman C/C++ mempunyai beberapa kelemahan dalam pemakaian tipe data string dan integer, namun bahasa ini cukup andal untuk membuat berbagai aplikasi perangkat lunak. Aspek yang perlu diperhatikan adalah kehati-hatian dalam memakai tipe data yang ada terutama string dan integer sehingga berbagai kerawanan dapat dihindari.

Disamping kerawanan yang ada pada tipe data string dan integer, tipe data yang lain seperti *array* bisa jadi mempunyai beberapa kelemahan. Untuk itu perlu ditelusuri lebih jauh berbagai kerawanan tipe data yang tidak dibahas disini.

## DAFTAR PUSTAKA

- Buchlovsky, Peter. Butcher, Adam. *Buffer Overflow Vulnerabilities ( Exploits and Desensive Techniques)*, University of Birmingham, UK, 2004
- Kadir, Abdul. *Pemrograman C++*, Penerbit Andi, Yogyakarta. 2003
- Seacord, Robert C, Rafail, John. *Best Pratices for Secure Coding*, Carnigie Mellon University / Software Enginering Institut, 2005  
[http://swerl.tudelft.nl/leon/cobassa2005/talks/seacord-best\\_practices\\_secure\\_coding.pdf](http://swerl.tudelft.nl/leon/cobassa2005/talks/seacord-best_practices_secure_coding.pdf) diakses tanggal 25 April 2007
- Seacord, Robert *Managed String Library for C*. Mr. Dobb Portal 2005  
<http://www.ddj.com/newsletters> diakses tanggal 25 April 2007
- TESO Security Group, *Exploiting Format String Vulnerabilities version 1*, 2001  
[www.team-teso.net](http://www.team-teso.net) diakses tanggal 25 April 2007
- Thuemmel, Andreas. *Analysis of Format String Bug*, Denmark, 2001  
<http://julianor.tripod.com/format-bug-analysis.pdf> diakses tanggal 27 April 2007
- Wheeler, David A. *Programming Secure Aplications for Unix-like System*, 2003  
<http://www.dwheeler.com/secure-programs> diakses tanggal
- Wheeler, David A. *Secure Programming for Linux and Unix HOWTO*, Copyright by David A Wheeler, 2003 <http://www.dwheeler.com/secure-programs> diakses tanggal 27 April 2007
- Xu, Haizhi. *Buffer Overflow Vulnerabilities and Counter Action*, 2004  
[web.syr.edu/~hxu02/mypapers/bufferoverflow.ppt](http://web.syr.edu/~hxu02/mypapers/bufferoverflow.ppt) diakses tanggal 17 Mei 2007