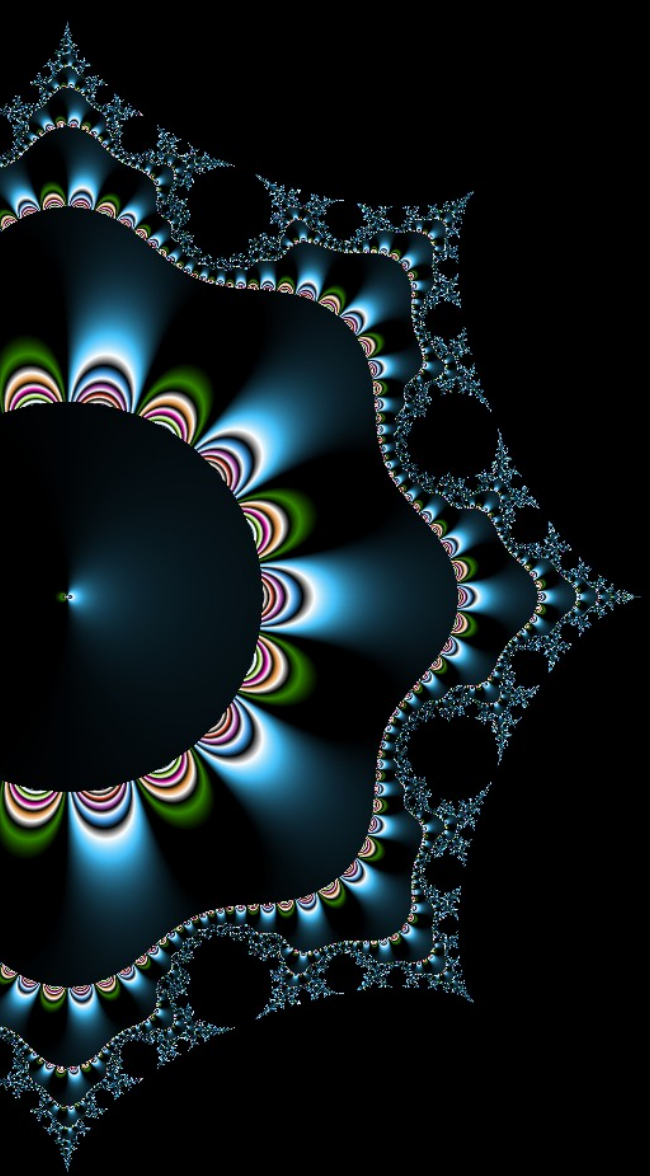
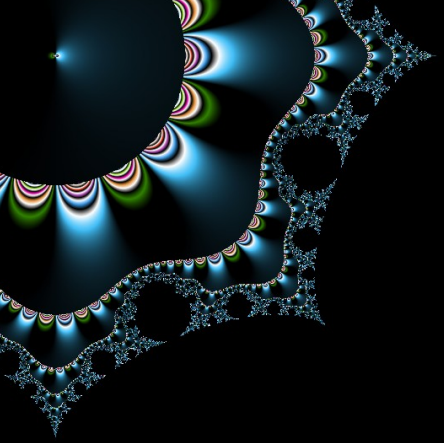
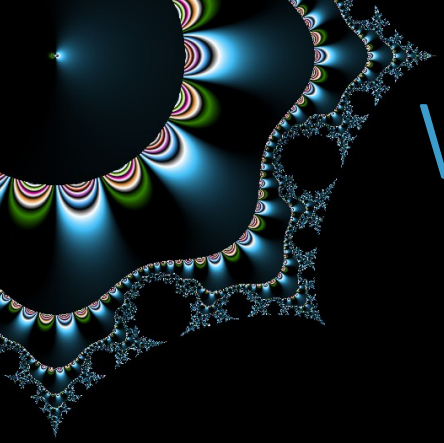


05b – Format String

Corinne HENIN
www.arsouyes.org



What’s the subject ?



What’s the subject ?

Format Strings

%s,%d, …

Exploitable

read/write in memory

Just a little less old

University of Wisconsin(1989)

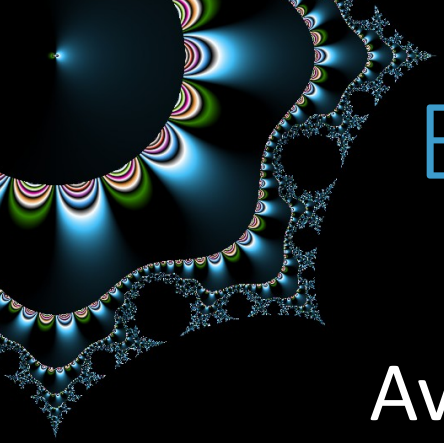
Discovery of the bug

ProFTPD(1999)

Discovery of a way to exploit

« Format Strings Attacks » by Tim Newsham (2000)

First documentation



But it’s still up to date too

Avaibility issues in HarmonyOs (huawei)

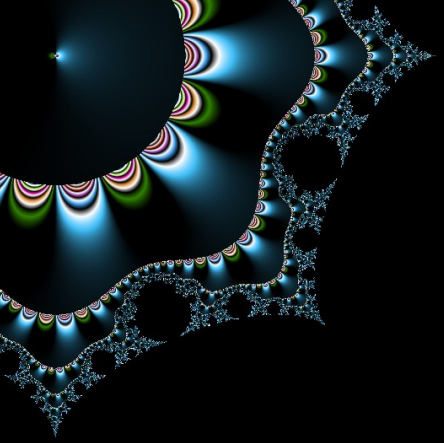
CVE-2022-31753

ASUS RT-AX88U remote code execution

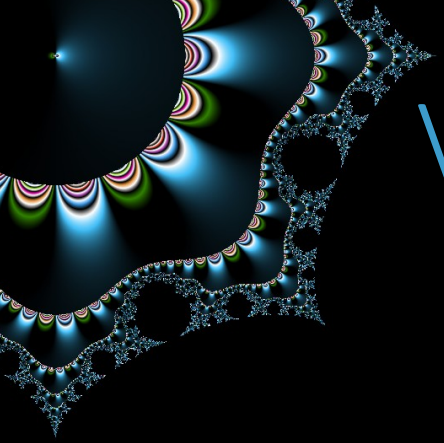
CVE-2022-26674

Fortinet FortiOS code execution

CVE-2024-23113



Printf & *cie*



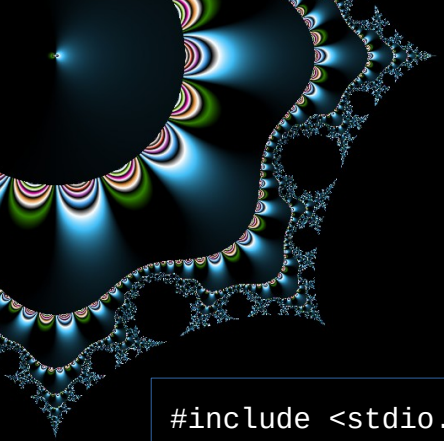
Variadic function

Infinite arity

Unlimited number of parameters

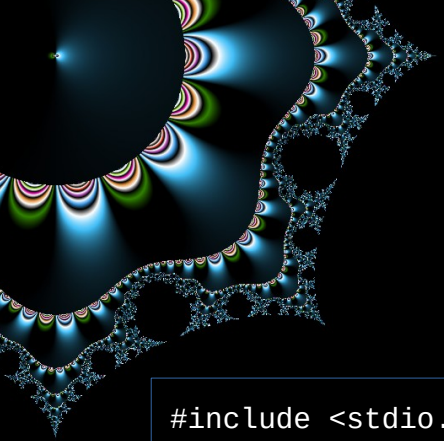
```
int func(type variable, ...);
```

C, C++



Sample Printf

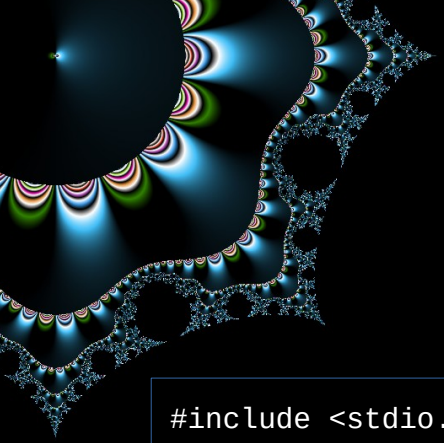
```
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```



Sample Printf

```
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

```
$ ./a.out 42
```



Sample Printf

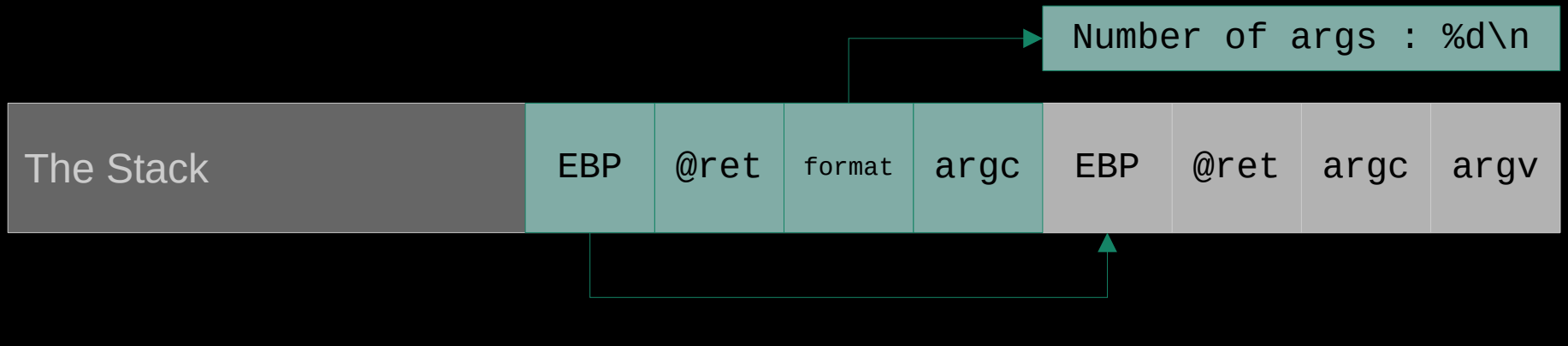
```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

```
$ ./a.out 42
```

Sample Printf

```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

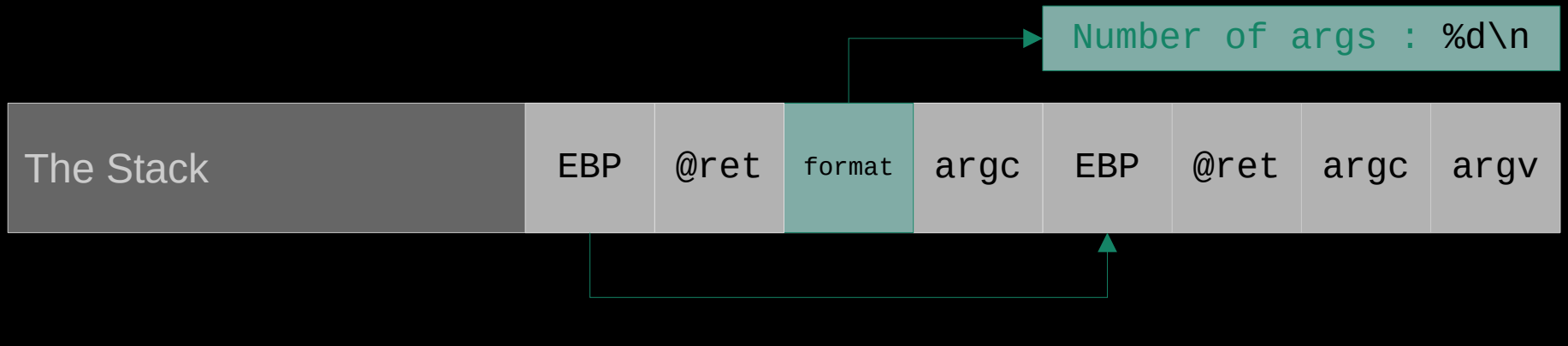
```
$ ./a.out 42
```



Sample Printf

```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

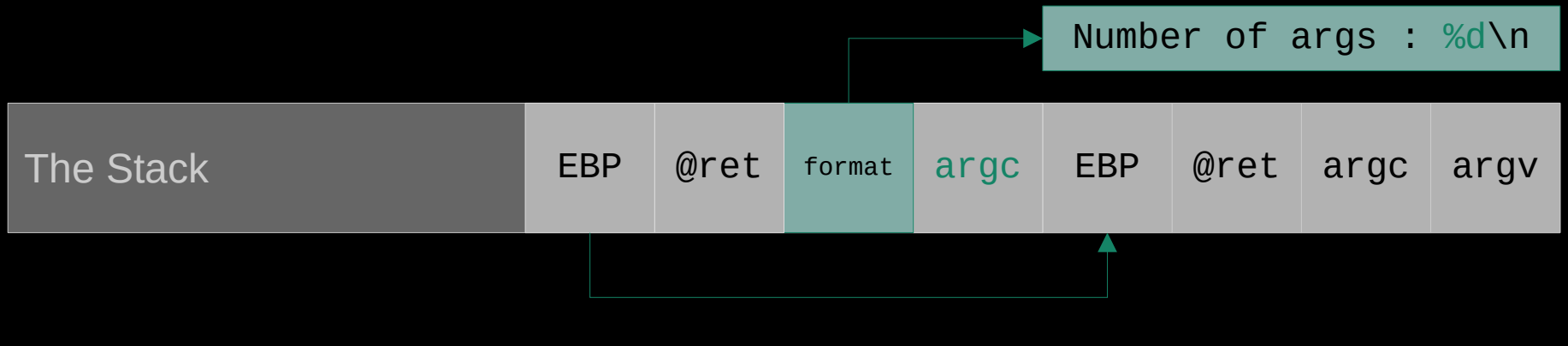
```
$ ./a.out 42
Number of args :
```

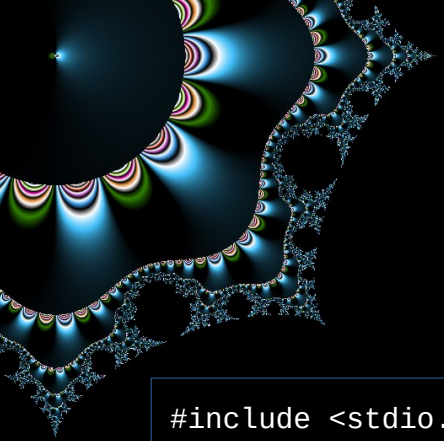


Sample Printf

```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

```
$ ./a.out 42
Number of args : 2
```





Sample Printf

```
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

```
$ ./a.out
```

Sample Printf

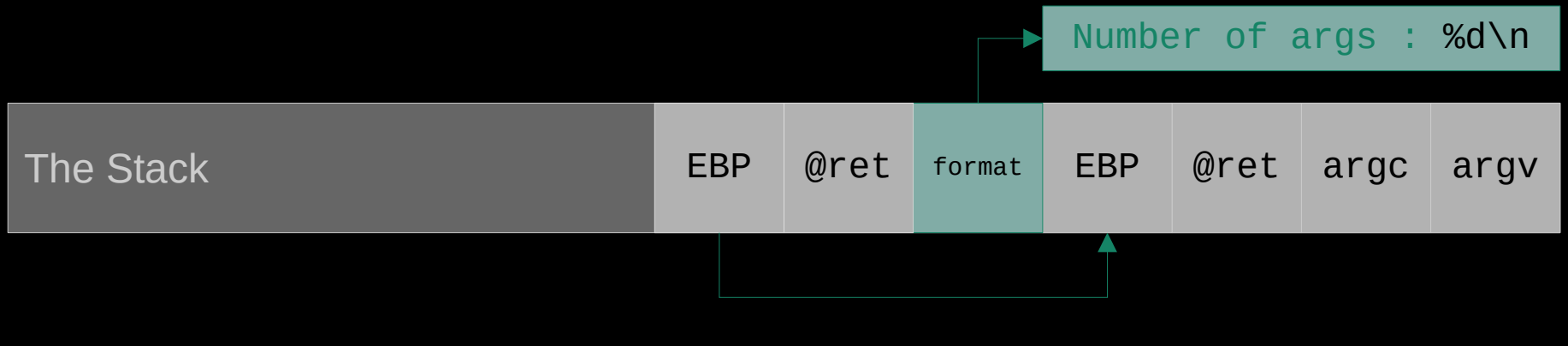
```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

```
$ ./a.out 42
```

Sample Printf

```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

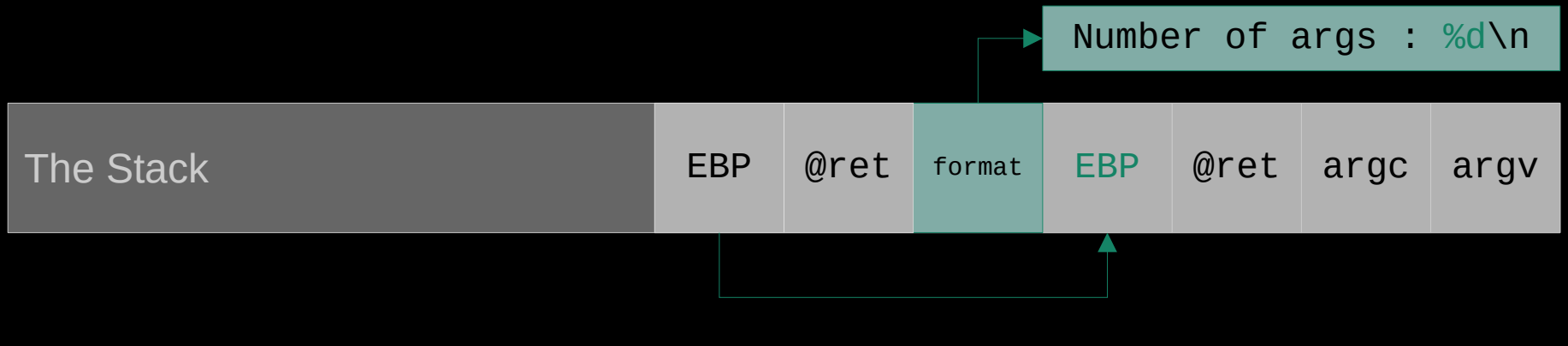
```
$ ./a.out 42
Number of args :
```

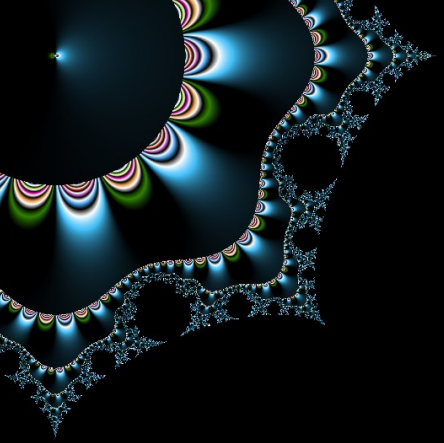


Sample Printf

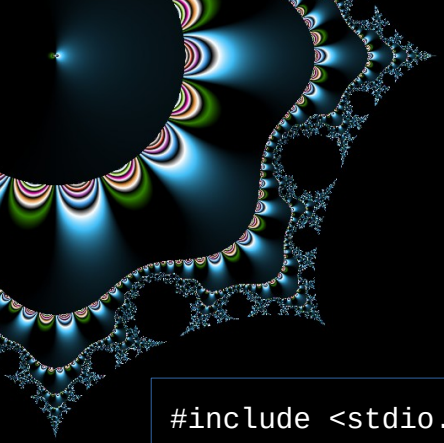
```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf("Number of args : %d\n", argc) ;
    return 0 ;
}
```

```
$ ./a.out 42
Number of args : -6472
```



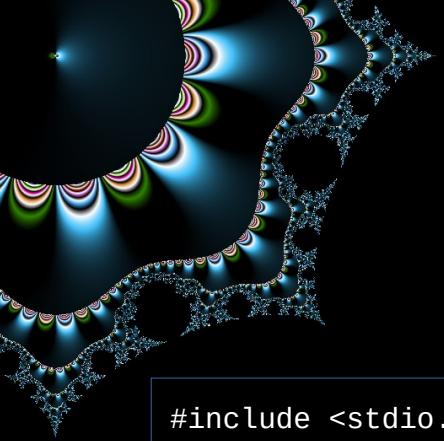


Vulnerable Code



Sample Printf

```
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```



Sample Printf

```
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out 12
```

Sample Printf

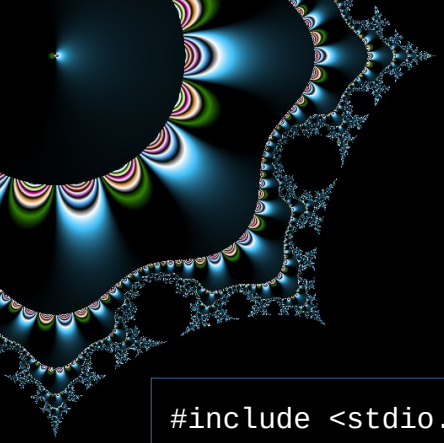
```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out 12
```

Sample Printf

```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out 12
12
```



Sample Printf

```
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out %x
```

Sample Printf

```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out %x
```

Sample Printf

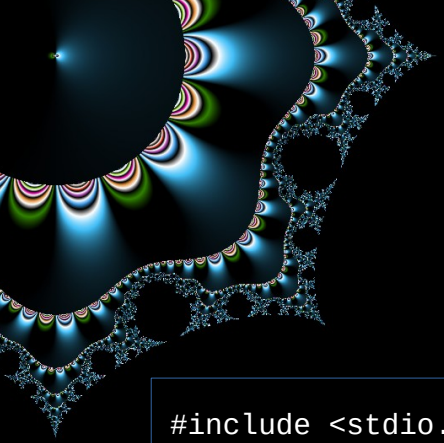
```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out %x
8f6d86a8
```

Sample Printf

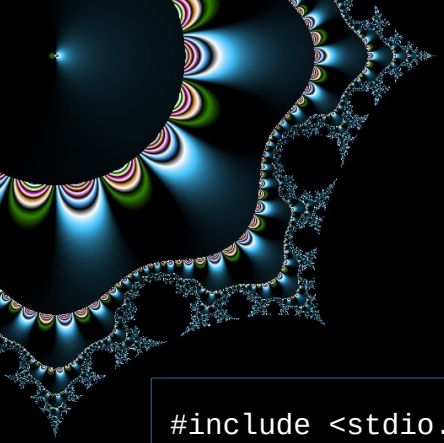
```c
#include <stdio.h>


int main(int argc, char ** argv) {
    printf(argv[1]) ;
    return 0 ;
}
```

```
$ ./a.out %s%s%s%s%s%s
Segmentation fault (core dumped)
```



Sample Printf

```
#include <stdio.h>
int main(int argc, char ** argv) {
    int secret = 42;
    printf(argv[1]) ;
    ...
}
```



Sample Printf

```
#include <stdio.h>
int main(int argc, char ** argv) {
    int secret = 42;
    printf(argv[1]) ;
    ...
}
```

```
$ ./a.out %d
```

Sample Printf

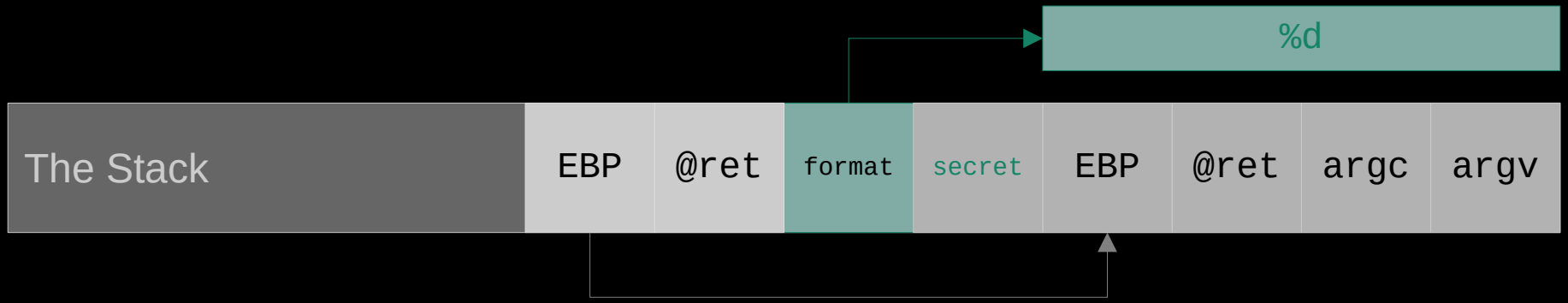
```c
#include <stdio.h>
int main(int argc, char ** argv) {
    int secret = 42;
    printf(argv[1]) ;
    ...
}
```

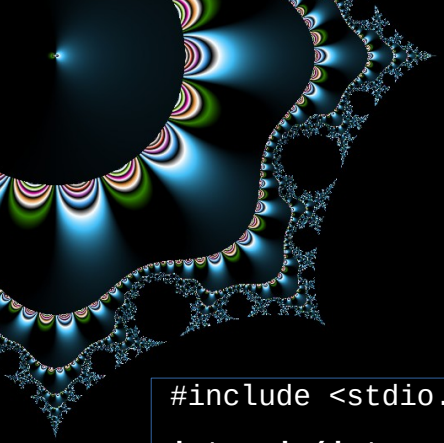
```
$ ./a.out %d
```

Sample Printf

```c
#include <stdio.h>
int main(int argc, char ** argv) {
    int secret = 42;
    printf(argv[1]) ;
    ...
}
```

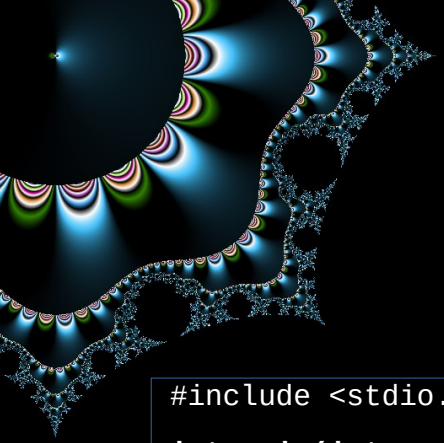
```
$ ./a.out %d
42
```





Sample Printf

```
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```



Sample Printf

```
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```

```
$ ./a.out 1%n
```

Sample Printf

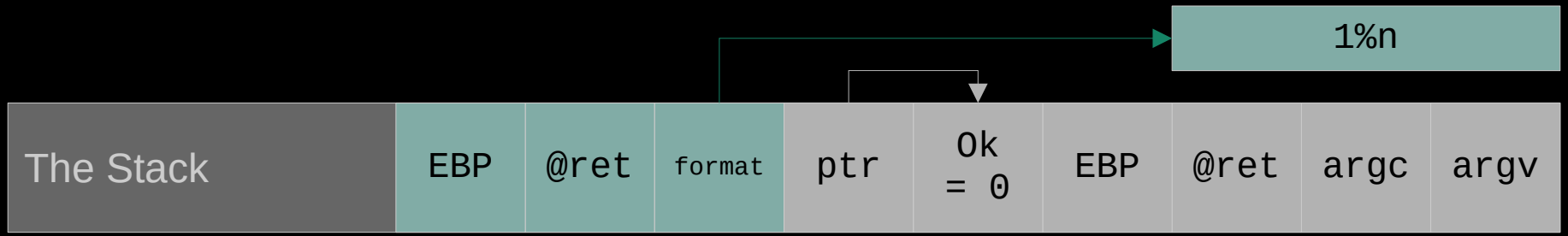
```c
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```

```
$ ./a.out 1%n
```

Sample Printf

```c
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```

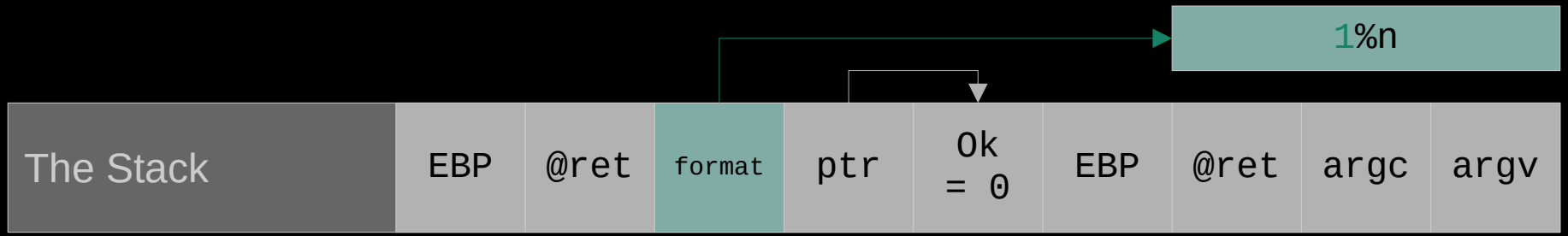
```
$ ./a.out 1%n
```



Sample Printf

```c
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```

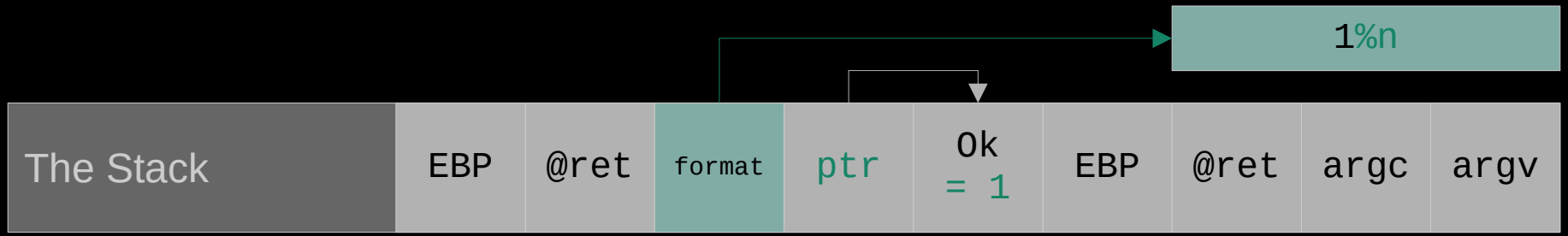
```
$ ./a.out 1%n
1
```

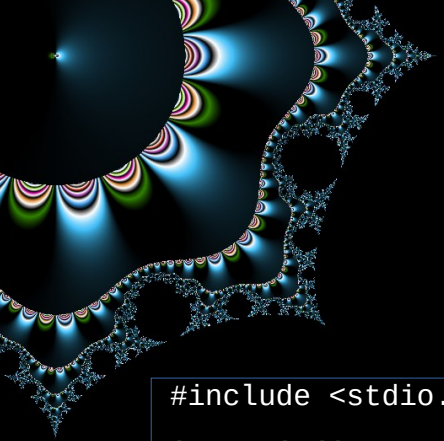


Sample Printf

```c
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```

```
$ ./a.out 1%n
1
```

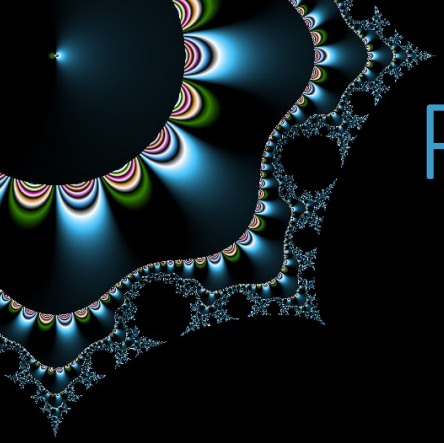




Sample Printf

```
#include <stdio.h>
int main(int argc, char ** argv) {
    int ok = 0 ;
    int * ptr = &ok ;
    printf(argv[1]) ;
    return ok ;
}
```

```
$ ./a.out 1%n
1
$ echo $?
1
```



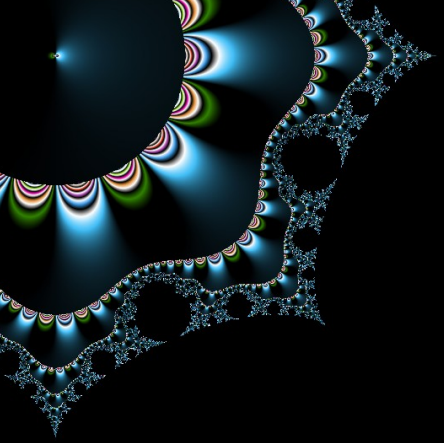
Possibilities

Read memory

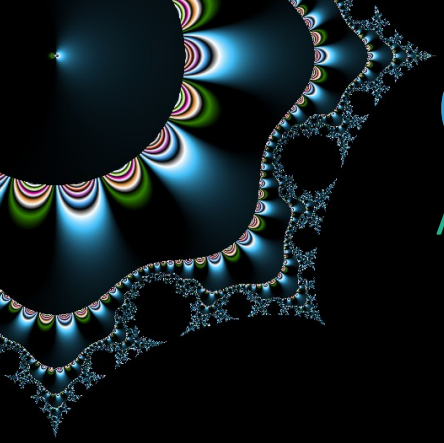
%x %s

Write memory

%n



Protection



Clean code

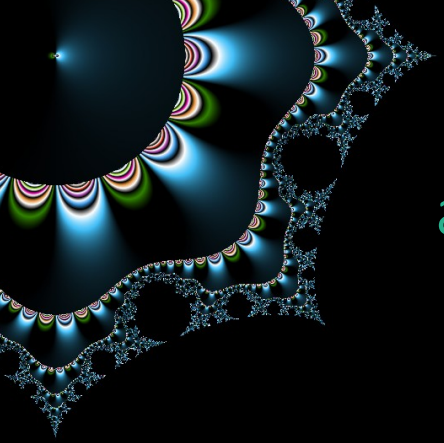
Avoid the problem

Always write format string

Especially when using users entries

Don’t use shortcuts

⚠ printf("%s", chaine) ≠ printf(chaine)



Defense in depth

a posteriori

Compilation options

(-Wformat -Wformat-nonliteral -Wformat-security -Wmissing-format-attribute)

But they are only warnings