

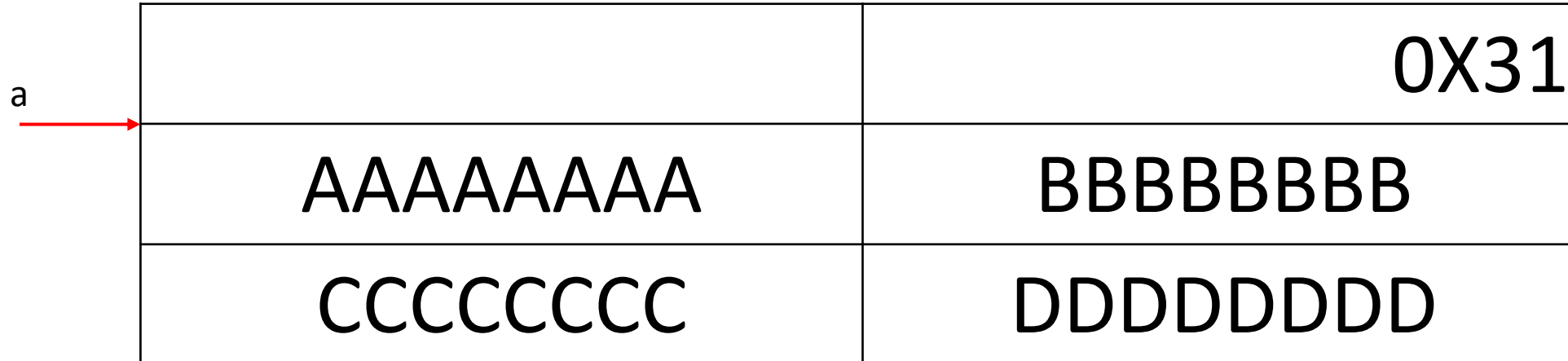
UAF

CyKor 정수환

<https://github.com/1nteger-c/uaf>

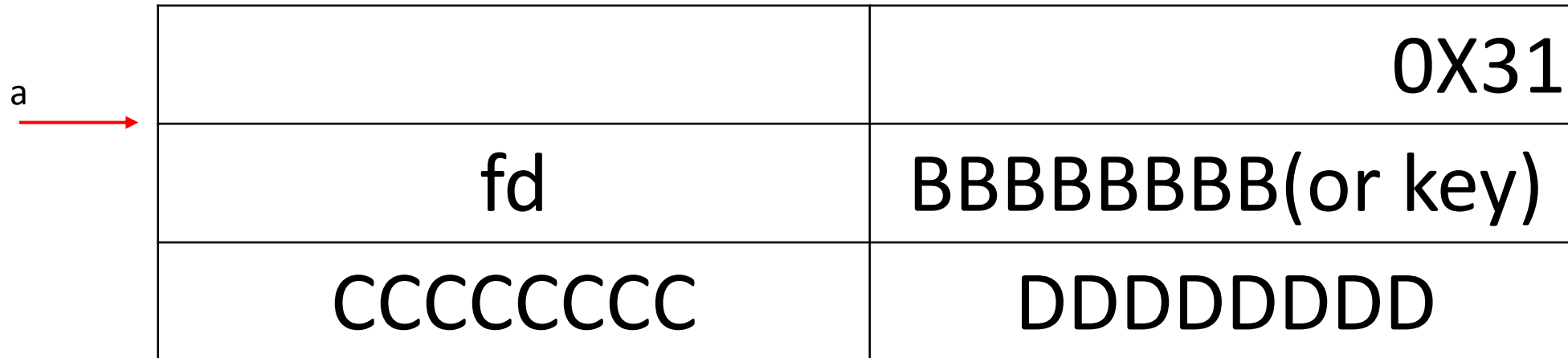
UAF (Use - After - Free)

a = malloc(0x20)



UAF (Use - After - Free)

`a = malloc(0x20) -> free()`

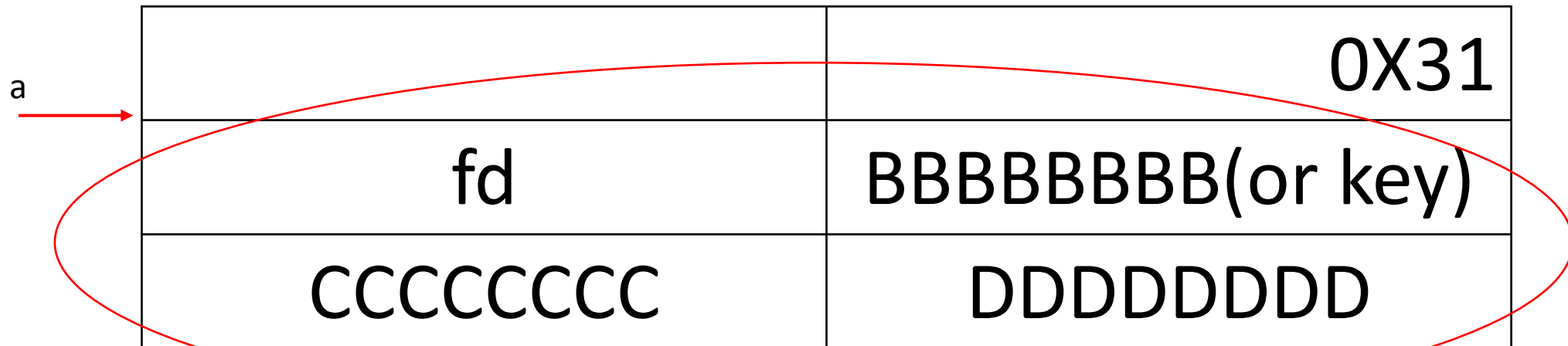


The diagram shows a memory block divided into six cells arranged in two rows and three columns. A red arrow labeled 'a' points to the first row. The first row contains an empty cell, the text '0X31', and the text 'BBBBBBBBBB(or key)'. The second row contains the text 'fd', the text 'CCCCCCCCC', and the text 'DDDDDDDDD'.

	0X31	BBBBBBBBBB(or key)
fd	CCCCCCCCC	DDDDDDDDD

UAF (Use - After - Free)

`a = malloc(0x20) -> free()`



데이터가 사라지지 않음!!!

UAF (Use - After - Free)

```
a = malloc(0x20) -> free()
```

여기서 malloc(0x20)을 하면??

CCCCCCCC

DDDDDDDDDD

데이터가 사라지지 않음!!!

UAF (Use - After - Free)

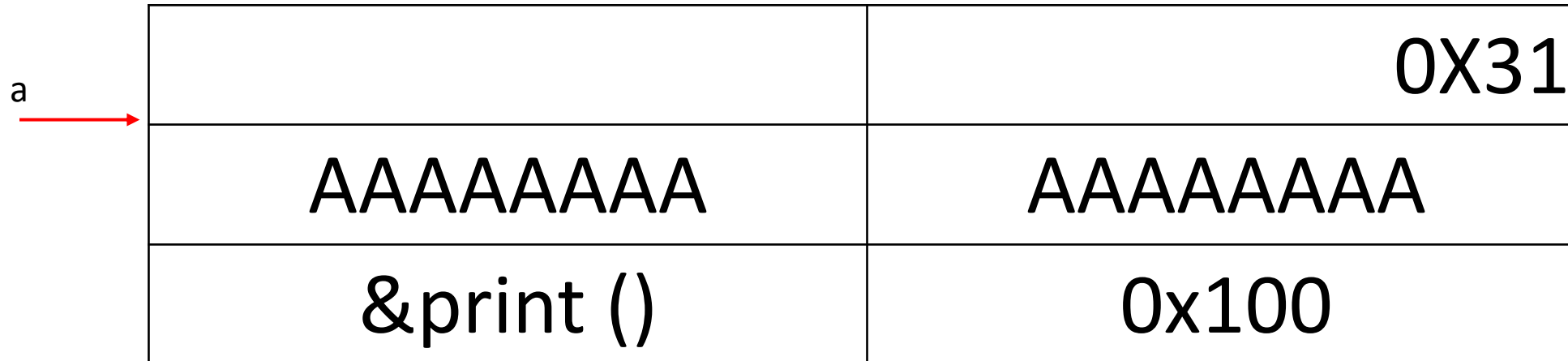
`a = malloc(0x20) -> free() -> b = malloc(0x20)` : 아래와 동일한 곳에 할당..!

a , b →		0X31
	fd	BBBBBBBBBB(or key)
	CCCCCCCCC	DDDDDDDDD

UAF (Use - After - Free)

```
struct Name{  
    char name[0x10];  
    void(*print_name)(void*);  
    score();  
}
```

```
Name * a = malloc(0x20)
```



	0X31
AAAAAAAAAA	AAAAAAAAAA
&print ()	0x100

UAF (Use - After - Free)

a = malloc(0x20) -> free() -> b = malloc(0x20)

	0X31
AAAAAAAA	AAAAAAAA
&system	0x100

UAF (Use - After - Free)

a = malloc(0x20) -> free() -> b = malloc(0x20)

a, b →		0X31
	AAAAAAAA	AAAAAAAA
	&system	0x100

만약, 여기서 a를 이용해서 print_name 함수를 호출한다면??

UAF (Use - After - Free)

a = malloc(0x20) -> free() -> b = malloc(0x20)

a, b →		0X31
	AAAAAAAA	AAAAAAAA
	&system	0x100

만약, 여기서 a를 이용해서 print_name 함수를 호출한다면?? → system 호출 가능..!

Double Free

- Free -> Free ??

```
a = malloc(0x30);  
free(a);  
free(a);
```

Double Free

- 16.04 (ERROR)

- `a = malloc(0x30);`
- `free(a);`
- `free(a);`

- 18.04 (OK)

- `a = malloc(0x30);`
- `free(a);`
- `free(a);`

- 20.04 (OK)

- `a = malloc(0x30);`
- `free(a);`
- `free(a);`

Double Free

- 16.04 (ERROR)

- `a = malloc(0x30);`
- `free(a);`
- `free(b);`
- `free(a);`

- 18.04 (OK)

- `a = malloc(0x30);`
- `free(a);`
- `free(a);`

- 20.04 (OK)

- `a = malloc(0x30);`
- `free(a);`
- `a의 key 자리 overwrite`
- `free(a);`

Tcache - dup

```
a = malloc(0x20);  
free(a);
```



tcache_bin : 0x1000

Tcache - dup

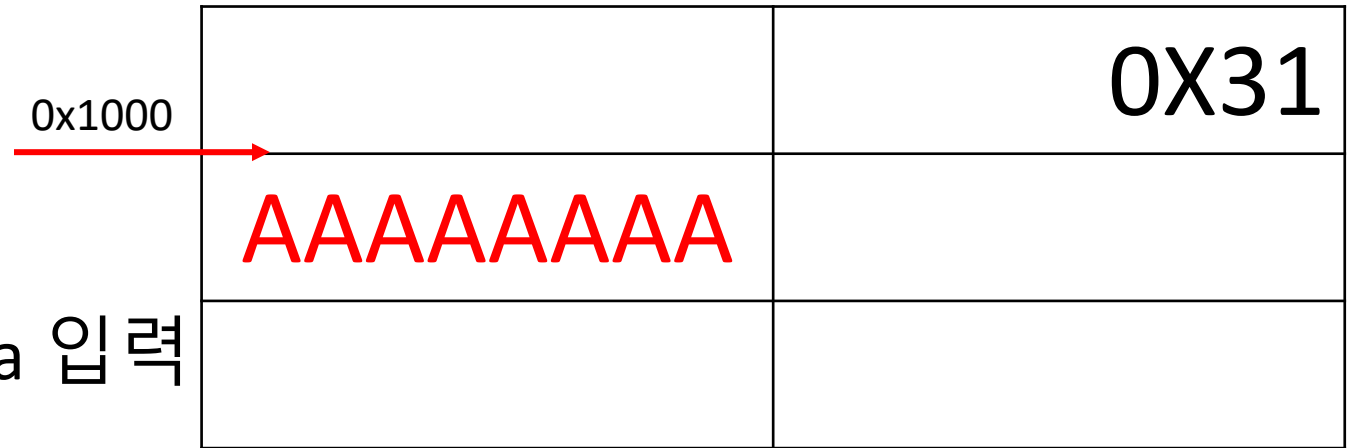
```
a = malloc(0x20);  
free(a);  
free(a);
```



tcache_bin : 0x1000 -> 0x1000

Tcache - dup

```
a = malloc(0x20);  
free(a);  
free(a);  
b = malloc(0x20); //0xaaaaaaaa 입력
```



tcache_bin : 0x1000 -> 0xAAAAAAAA

Tcache - dup

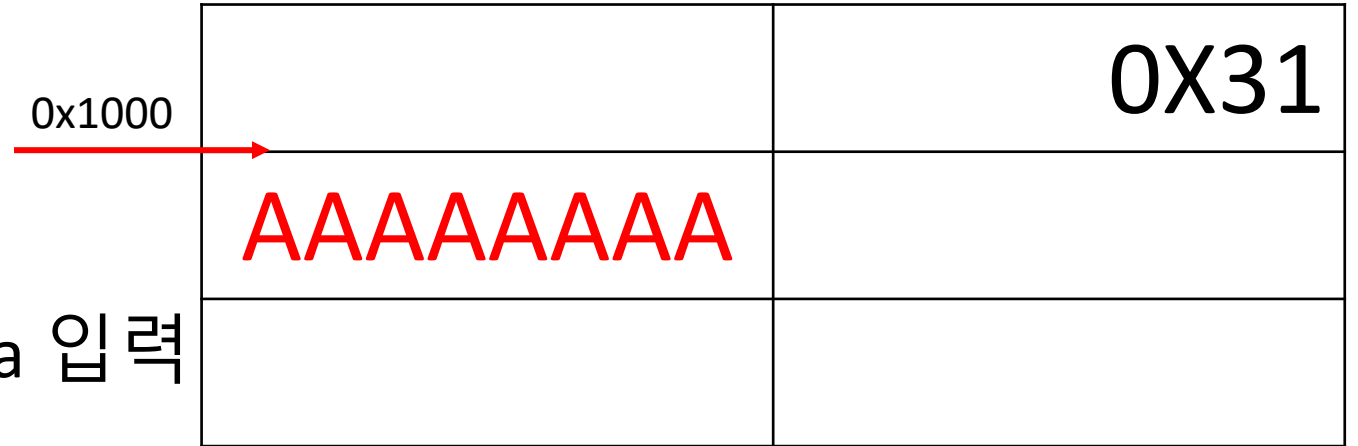
```
a = malloc(0x20);
```

```
free(a);
```

```
free(a);
```

```
b = malloc(0x20); //0xaaaaaaaa 입력
```

```
c = malloc(0x20);
```



tcache_bin : 0xAAAAAAAA

Tcache - dup

```
a = malloc(0x20);
```

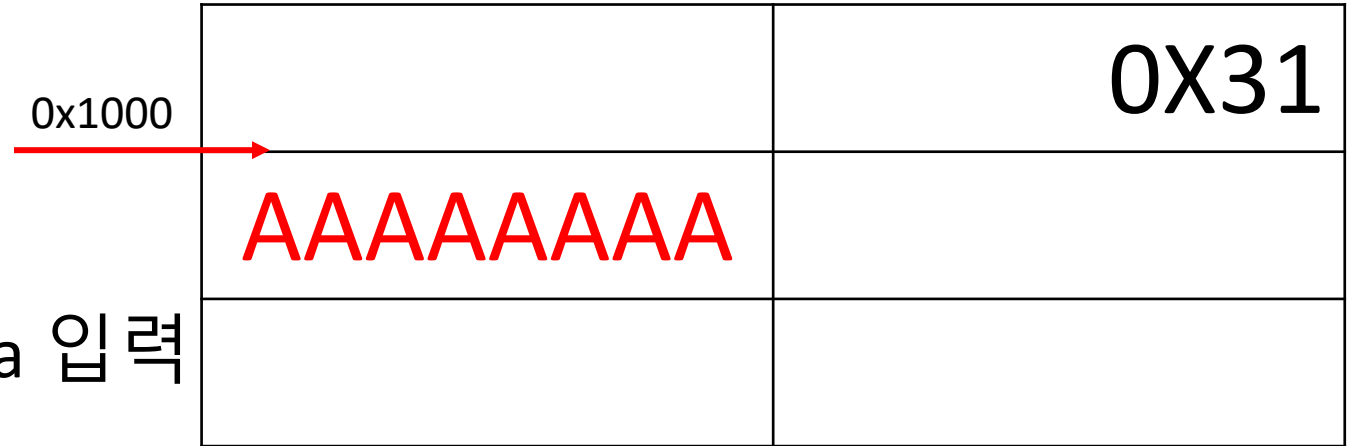
```
free(a);
```

```
free(a);
```

```
b = malloc(0x20); //0xaaaaaaaa 입력
```

```
c = malloc(0x20);
```

```
d = malloc(0x20); // 0xAAAAAAAA에 chunk 할당..!
```



tcache_bin :

- 어디에 ? 무엇으로 ? 덮어야 할까요

Hook (__malloc_hook / __free_hook)

oneshot_gadget

과제

- git에 함께 첨부되어 있는 homework/tcache
18.04 / 20.04 에서 각각 Exploit 해보기

(18.04가 더 쉬우니 18.04먼저 하세요)

제출 메일 : pk2861@naver.com 기한 : 12/1 23:59

질문있으면 갠톡하세요