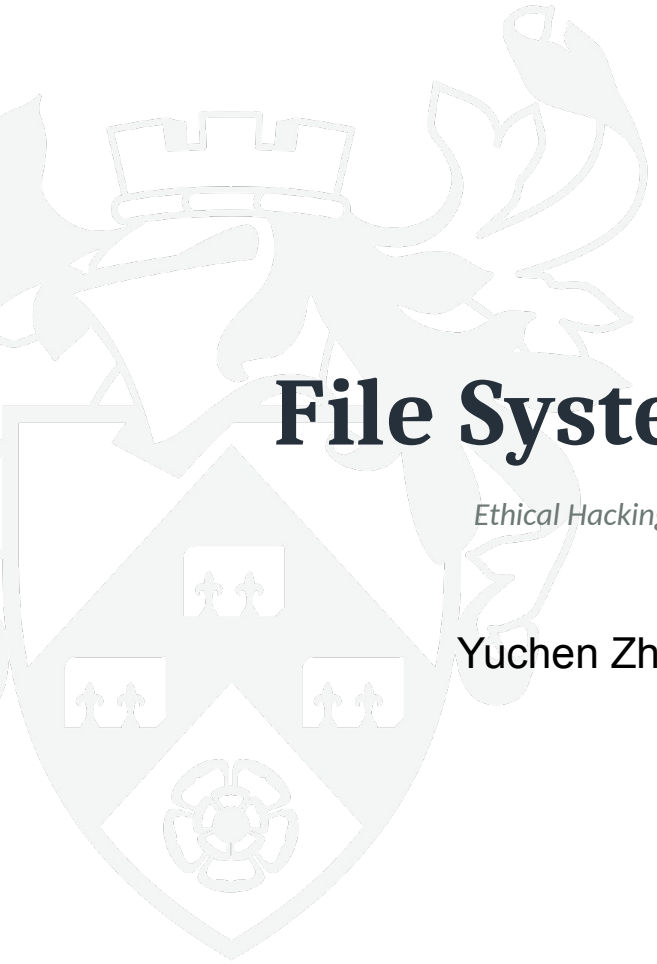


UNIVERSITY of York

File System API and Monitoring

Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

Outline

- File System API
  - Creating Files and Directories
  - Reading and Writing Files
- Monitoring System Calls
- Further Reading

Files

File: a linear array of bytes, each of which you can read or write.

Each file has a **user-readable name** (e.g. “foo”).

Each file has some kind of **low-level name** (usually a number, e.g. 10).

- is often referred to as its **inode number** (**i-number**).
- often, the user is not aware of this name.

Directories

Directory:

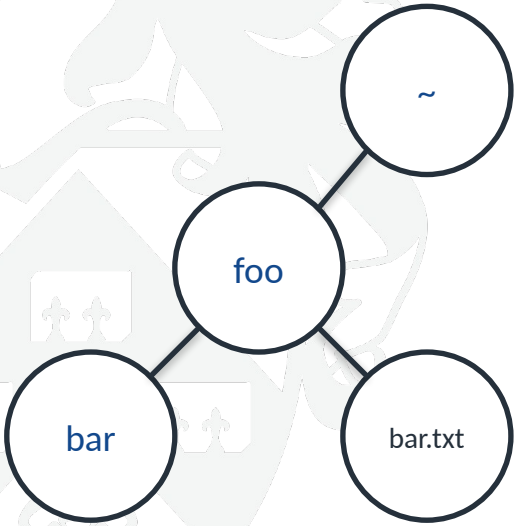
- like a file, also has a low-level name (i.e., an inode number)
- contains a list of (user-readable name, low-level name) pairs.

For example, let's say there is a file with the low-level name "10", and it is referred to by the user-readable name of "foo".

The directory that "foo" resides in would have an entry ("foo", "10") that maps the user-readable name to the low-level name.

Each entry in a directory refers to either files or other directories. By placing directories within other directories, users are able to build a **directory tree** (or **directory hierarchy**), under which all files and directories are stored.

Directory Tree: Example



Home directory of current user
/home/ub

```
ub@ub:~/foo$ ls
ub@ub:~/foo$ mkdir bar
ub@ub:~/foo$ touch bar.txt
ub@ub:~/foo$ ls
bar  bar.txt
ub@ub:~/foo$ ls /home/ub/foo
bar  bar.txt
ub@ub:~/foo$
```

Directory Tree: Example

We could refer to the file by its **absolute pathname**,

valid directories: /, /foo, /bar, /bar/bar, and /bar/foo

valid files: /foo/bar.txt and /bar/foo/bar.txt.

Directories and files can have the **same name** as long as they are **in different locations** in the file-system tree

- (e.g., there are two files named bar.txt in the figure, /foo/bar.txt and /bar/foo/bar.txt).

Creating Files

Open | create_file.c ~/foo | Save

```
1 #include <fcntl.h>
2
3 int main(){
4
5 int fd = open("foo", O_CREAT|O_WRONLY|O_TRUNC, S_IRUSR|S_IWUSR);
6
7
8 return 0;
9 }
10
```

Macros from fcntl.h

Macros from sys/stat.h

00100 00001 01000 0400 0200

Octal numbers (base 8)

```
ub@ub:~/foo$ ls
create_file.c
ub@ub:~/foo$ gcc -o create_file create_file.c
ub@ub:~/foo$ ls
create_file  create_file.c
ub@ub:~/foo$ ./create_file
ub@ub:~/foo$ ls
create_file  create_file.c  foo
ub@ub:~/foo$
```

Creating Files

- The routine **open()** takes a number of different flags.
- The second parameter creates the file (O_CREAT) **if it does not exist**, ensures that the file can **only be written to** (O_WRONLY), and, if the file already exists, truncates it to a size of zero bytes thus **removing** any existing content (O_TRUNC).
- The third parameter specifies **permissions**, in this case making the file readable and writable by the owner.

Reading and Writing Files

```
ub@ub:~/foo$ ls
create_file  create_file.c  foo
ub@ub:~/foo$ echo hello > foo
ub@ub:~/foo$ cat foo
hello
ub@ub:~/foo$
```

Short for con**cat**enate,
used to read, display, and concatenate text files.

Monitoring System Calls: strace

```
ub@ub:~/foo$ strace cat foo
execve("/usr/bin/cat", ["cat", "foo"], 0x7ffdc4cc8ef8 /* 46 vars */) = 0
brk(NULL)                               = 0x555d3906c000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffbfb1ced0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f5e354b7000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=60987, ...}, AT_EMPTY_PATH) = 0
openat(AT_FDCWD, "foo", O_RDONLY)       = 3
newfstatat(3, "", {st_mode=S_IFREG|0600, st_siz
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PR
read(3, "hello\n", 131072)              = 6
write(1, "hello\n", 6hello
)                       = 6
read(3, "", 131072)                     = 0
munmap(0x7f5e35483000, 139264)          = 0
close(3)                                = 0
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

strace

strace: powerful diagnostic, debugging, and instructional utility for Linux.

- Used to monitor the system calls and signals that a program receives, allowing you to inspect the behaviour of a program and troubleshoot issues related to system calls
- Intercepts and records the system calls that are made by the program and the signals that are received by the program.
- Useful for understanding how a program interacts with the OS, identifying performance bottlenecks, diagnosing errors, and debugging issues such as improper file access or permission problems.

Creating Files: strace

Open | create_file.c ~/foo | Save

```c
#include <fcntl.h>

int main(){

int fd = open("foo", O_CREAT|O_WRONLY|O_TRUNC, S_IRUSR|S_IWUSR);


return 0;
}
```

```
mprotect(0x7f1d0fb82000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=
 = 0
munmap(0x7f1d0fb39000, 60987)           = 0
openat(AT_FDCWD, "foo", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

System calls for file handling: Open() and Close()

open(): This system call is used to open a file or device. It takes a filename and various flags as arguments and returns a file descriptor, which is an integer representing the opened file. This file descriptor is then used in subsequent read and write operations.

close(): This system call is used to close an open file descriptor when it is no longer needed. It takes the file descriptor as an argument and releases any resources associated with it. Once a file descriptor is closed, it cannot be used for further read or write operations.

System calls for file handling: Read() and Write()

read(): This system call is used to read data from an open file descriptor into a buffer in memory. It takes the file descriptor, a buffer where the data will be stored, and the number of bytes to read as arguments. It returns the number of bytes actually read, which may be less than the requested amount if the end of the file is reached or if an error occurs.

write(): This system call is used to write data from a buffer in memory to an open file descriptor. It takes the file descriptor, a buffer containing the data to be written, and the number of bytes to write as arguments. It returns the number of bytes actually written, which may be less than the requested amount if there is not enough space on the device or if an error occurs.

Open | write_to_file.c ~/foo | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int main(){

int fd;
char text[] = "Hello World\n";

fd=open("output.txt",O_WRONLY|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR);

write(fd,text,sizeof(text));


return 0;
}
```

```
ub@ub:~/foo$ gcc -o write_to_file write_to_file.c
ub@ub:~/foo$ ./write_to_file
ub@ub:~/foo$ ls -l
total 48
-rwxrwxr-x 1 ub ub 15968 Feb 26 13:34 create_file
-rw-rw-r-- 1 ub ub   113 Feb 26 13:31 create_file.c
-rw------- 1 ub ub     6 Feb 26 16:09 foo
-rw------- 1 ub ub    13 Feb 26 22:40 output.txt
-rwxrwxr-x 1 ub ub 16064 Feb 26 22:39 write_to_file
-rw-rw-r-- 1 ub ub   239 Feb 26 22:39 write_to_file.c
ub@ub:~/foo$ cat output.txt
Hello World
ub@ub:~/foo$
```

Open | write_to_file.c ~/foo | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int main(){

int fd;
char text[] = "Hello World\n";

fd=open("output.txt",O_WRONLY|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR);

write(fd,text,sizeof(text));


return 0;
}
```

```
munmap(0x7f8bddcf1000, 60987)           = 0
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
write(3, "Hello World\n\0", 13)         = 13
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

Using Offsets

System Calls	Return Code	Current Offset
`fd = open("file", O_RDONLY);`	3	0
`read(fd, buffer, 100);`	100	100
`read(fd, buffer, 100);`	100	200
`read(fd, buffer, 100);`	100	300
`read(fd, buffer, 100);`	0	300
`close(fd);`	0	–

Using Offsets

- A process that **opens a file** of size 300 bytes
- Reads it by calling the **read()** system call repeatedly, each time reading 100 bytes.
- Trace of the relevant system calls, along with the **values** returned by each system call, and the value of the **current offset** in the Open File Table (OFT) for this file access.

Using Offsets: Two File Descriptors

System Calls	Return Code	OFT[10] Current Offset	OFT[11] Current Offset
`fd1 = open("file", O_RDONLY);`	3	0	–
`fd2 = open("file", O_RDONLY);`	4	0	0
`read(fd1, buffer1, 100);`	100	100	0
`read(fd2, buffer2, 100);`	100	100	100
`close(fd1);`	0	–	100
`close(fd2);`	0	–	–

Using Offsets: Two File Descriptors

- a process opens the same file twice and issues a read to each of them.
- two file descriptors are allocated (3 and 4), and each refers to a different entry in the open file table
- in this example, entries 10 and 11
  - as shown in the table heading OFT (Open File Table).
- each current offset is updated independently.

Open File Table (OFT)

The Open File Table (OFT): used by the OS to manage **open files** within a process. Contains entries for each file that is currently open by a process, e.g.:

- **File descriptor:** used by the process to refer to the file in subsequent read, write, or close operations.
- **File pointer:** A pointer indicating the current position within the file. Updated as the process reads from or writes to the file.
- **File access mode:** Information about the access mode in which the file was opened (e.g., read-only, write-only, or read-write).

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#define FILE_SIZE 50
#define CHUNK_SIZE 10

int main()
{
    char text[] = "This is some text written to a file of 50 bytes.  ";
    char buffer[CHUNK_SIZE + 1]; //+1 for null terminator

    int fd = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    int bytes_written = write(fd, text, sizeof(text) - 1);
    close(fd);

    fd = open("output.txt", O_RDONLY);
    printf("Reading file contents: \n");
    int total_bytes_read = 0;
    ssize_t bytes_read;
    while ((bytes_read = read(fd, buffer, CHUNK_SIZE)) > 0)
    {
        buffer[bytes_read] = '\0'; // null terminate the buffer
        printf("%s\n", buffer);
        total_bytes_read += bytes_read;
    }
    close(fd);
    printf("\nTotal bytes read: %d\n", total_bytes_read);
}
```

```
yuchen@hp-elitebook:~/test$ gcc -o read_from_file read_from_file.c
yuchen@hp-elitebook:~/test$ ./read_from_file
Reading file contents:
This is so
me text wr
itten to a
 file of 5
0 bytes.

Total bytes read: 50
yuchen@hp-elitebook:~/test$ cat output.txt
This is some text written to a file of 50 bytes.  yuchen@hp-elitebook
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#define FILE_SIZE 50
#define CHUNK_SIZE 10

int main()
{
    char text[] = "This is some text written to a file of 50 bytes.  ";
    char buffer[CHUNK_SIZE + 1]; //+1 for null terminator

    int fd = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    int bytes_written = write(fd, text, sizeof(text) - 1);
    close(fd);

    fd = open("output.txt", O_RDONLY);
    printf("Reading file contents: \n");
    int total_bytes_read = 0;
    ssize_t bytes_read;
    while ((bytes_read = read(fd, buffer, CHUNK_SIZE)) > 0)
    {
        buffer[bytes_read] = '\0'; // null terminate the buffer
        printf("%s\n", buffer);
        total_bytes_read += bytes_read;
    }
    close(fd);
    printf("\nTotal bytes read: %d\n", total_bytes_read);
}
```

1 is for standard output

```
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 020060) = 3
write(3, "This is some text written to a f"..., 50) = 50
close(3)                                = 0
openat(AT_FDCWD, "output.txt", O_RDONLY) = 3
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
getrandom("\xcc\x4a\x67\x37\x64\x53\xb7\x5a", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55929d0fe000
brk(0x55929d11f000)                     = 0x55929d11f000
write(1, "Reading file contents: \n", 24Reading file contents:
) = 24
read(3, "This is so", 10)               = 10
write(1, "This is so\n", 11This is so
)           = 11
read(3, "me text wr", 10)               = 10
write(1, "me text wr\n", 11me text wr
)           = 11
read(3, "itten to a", 10)               = 10
write(1, "itten to a\n", 11itten to a
)           = 11
read(3, " file of 5", 10)               = 10
write(1, " file of 5\n", 11 file of 5
)           = 11
read(3, "0 bytes.  ", 10)               = 10
write(1, "0 bytes.  \n", 110 bytes.
)           = 11
read(3, "", 10)                         = 0
close(3)                                = 0
write(1, "\n", 1
)                       = 1
write(1, "Total bytes read: 50\n", 21Total bytes read: 50
)  = 21
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

strace: Examples

```
openat(AT_FDCWD, "./foo/output.txt", O_RDONLY) = 3
newfstatat(3, "", {st_mode=S_IFREG|0600, st_size=13,
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PRIVATE
read(3, "Hello World\n\0", 131072)      = 13
write(1, "Hello World\n\0", 13Hello World
)         = 13
read(3, "", 131072)                     = 0
munmap(0x7f995288a000, 139264)          = 0
close(3)                                = 0
close(1)                                = 0
close(2)                                = 0
```

What could the C code be?

```
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
write(3, "Hello World\n\0", 13)         = 13
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

```
openat(AT_FDCWD, "./foo/output.txt", O_RDONLY) = 3
newfstatat(3, "", {st_mode=S_IFREG|0600, st_size=13,
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PRIVATE
read(3, "Hello World\n\0", 131072)      = 13
write(1, "Hello World\n\0", 13Hello World
)         = 13
read(3, "", 131072)                     = 0
munmap(0x7f995288a000, 139264)          = 0
close(3)                                = 0
close(1)                                = 0
close(2)                                = 0
```

Open | write_to_file.c ~/foo | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int main(){

int fd;
char text[] = "Hello World\n";

fd=open("output.txt",O_WRONLY|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR);

write(fd,text,sizeof(text));


return 0;
}
```

```
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
write(3, "Hello World\n\0", 13)         = 13
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

strace: Example

What could the Linux command be?

```
close(3)                                = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
write(1, "create_file  create_file.c  foo "..., 76create_file  create_file.c  foo  output.txt
e.c
) = 76
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

Read or Write to a Specific Offset

System Calls	Return Code	Current Offset
`fd = open("file", O_RDONLY);`	3	0
`lseek(fd, 200, SEEK_SET);`	200	200
`read(fd, buffer, 50);`	50	250
`close(fd);`	0	–

- The **`lseek()`** call first sets the current **offset** to 200.
- The subsequent **`read()`** then reads the next 50 bytes, and updates the current offset accordingly.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#define FILE_SIZE 50
#define CHUNK_SIZE 10

int main(){

char text[] = "This is some text written to a file of 50 bytes.  ";
char buffer[CHUNK_SIZE+1]; //+1 for null terminator
int fd = open("output.txt",O_WRONLY| O_CREAT | O_TRUNC);
int bytes_written = write(fd,text,sizeof(text)-1);  close(fd);

fd = open("output.txt",O_RDONLY);
off_t offset = CHUNK_SIZE *2;
lseek(fd,offset,SEEK_SET);

ssize_t bytes_read;
bytes_read = read(fd,buffer,CHUNK_SIZE);
buffer[bytes_read] = '\0'; // Null terminate the buffer
printf("Contents of the 3rd buffer: %s\n",buffer);

close(fd);

return 0;
}
```

0, denotes the starting of the file

```
ub@ub:~/foo$ ./read_from_file_lseek
Contents of the 3rd buffer: itten to a
ub@ub:~/foo$
```

Read or Write to a Specific Offset

```
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 020060) = 3
write(3, "This is some text written to a f"..., 50) = 50
close(3)                                = 0
openat(AT_FDCWD, "output.txt", O_RDONLY) = 3
lseek(3, 20, SEEK_SET)                  = 20
read(3, "itten to a", 10)               = 10
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
getrandom("\x40\x8b\xa1\x23\x8a\xe0\x5d\x32", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x556be1939000
brk(0x556be195a000)                     = 0x556be195a000
write(1, "Contents of the 3rd buffer: itte"..., 39Contents of the 3rd buffer: itten to a
) = 39
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~/foo$
```

Getting Information about Files: stat

`stat`: displays detailed information about a file or file system, such as its size, permissions, inode number, last access time, last modifications time.

- Provides a comprehensive overview of the **metadata** associated with a file or directory.

Getting Information about Files: stat

```
ub@ub:~/foo3$ mkdir foo
ub@ub:~/foo3$ touch foo.txt
ub@ub:~/foo3$ echo "hello" > foo.txt
ub@ub:~/foo3$ cp foo.txt foo/foo2.txt
ub@ub:~/foo3$ stat foo
  File: foo
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 803h/2051d      Inode: 659326      Links: 2
Access: (0775/drwxrwxr-x)  Uid: ( 1000/      ub)   Gid: ( 1000/      ub)
Access: 2024-02-27 20:05:12.083226590 +0000
Modify: 2024-02-27 20:05:06.558145972 +0000
Change: 2024-02-27 20:05:06.558145972 +0000
 Birth: 2024-02-27 20:04:06.089771168 +0000
ub@ub:~/foo3$ stat foo/foo2.txt
  File: foo/foo2.txt
  Size: 6               Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d      Inode: 671973      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/      ub)   Gid: ( 1000/      ub)
Access: 2024-02-27 20:05:06.558145972 +0000
Modify: 2024-02-27 20:05:06.558145972 +0000
Change: 2024-02-27 20:05:06.558145972 +0000
 Birth: 2024-02-27 20:05:06.558145972 +0000
ub@ub:~/foo3$
```

The stat structure

```
struct stat {
  dev_t     st_dev;     // ID of device containing file
  ino_t     st_ino;     // inode number
  mode_t    st_mode;    // protection
  nlink_t   st_nlink;   // number of hard links
  uid_t     st_uid;     // user ID of owner
  gid_t     st_gid;     // group ID of owner
  dev_t     st_rdev;    // device ID (if special file)
  off_t     st_size;    // total size, in bytes
  blksize_t st_blksize; // blocksize for filesystem I/O
  blkcnt_t  st_blocks;  // number of blocks allocated
  time_t    st_atime;   // time of last access
  time_t    st_mtime;   // time of last modification
  time_t    st_ctime;   // time of last status change
};
```

Getting Information about Files: stat() system call

Open | mystat.c ~/foo2 | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <file>\n", argv[0]);
        return 1;
    }

    const char *file_name = argv[1];
    struct stat file_stat;

    if (stat(file_name, &file_stat) == -1) {
        fprintf(stderr, "Error: %s\n", strerror(errno));
        return 1;
    }

    printf("File: %s\n", file_name);
    printf("Size: %ld bytes\n", file_stat.st_size);
    printf("Blocks allocated: %ld\n", file_stat.st_blocks);
    printf("Reference count: %ld\n", file_stat.st_nlink);

    return 0;
}
```

```
ub@ub:~/foo2$ gcc -o mystat mystat.c
ub@ub:~/foo2$ touch foo.txt
ub@ub:~/foo2$ echo "some text" > foo.txt
ub@ub:~/foo2$ ./mystat foo.txt
File: foo.txt
Size: 10 bytes
Blocks allocated: 8
Reference count: 1
ub@ub:~/foo2$
```

Making Directories

```
ub@ub:~$ mkdir foo2
ub@ub:~$ cd foo2
ub@ub:~/foo2$ ls
ub@ub:~/foo2$ ls -l
total 0
ub@ub:~/foo2$ ls -la
total 8
drwxrwxr-x  2 ub ub 4096 Feb 27 11:06 .
drwxr-x--- 24 ub ub 4096 Feb 27 11:06 ..
ub@ub:~/foo2$
```

```
ub@ub:~$ mkdir foo2
mkdir: cannot create directory ‘foo2’: File exists
ub@ub:~$ rmdir foo2
ub@ub:~$ mkdir foo2
ub@ub:~$
```

```
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=
mmap(NULL, 5712208, PROT_READ, MAP_PRIVATE, 3, 0)
close(3)                                = 0
mkdir("foo2", 0777)                     = 0
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
ub@ub:~$
```

```
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=5712208, ...}, AT_EMPTY_PA
mmap(NULL, 5712208, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff0cd400000
close(3)                                = 0
mkdir("foo2", 0777)                     = -1 EEXIST (File exists)
openat(AT_FDCWD, "/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2996, ...}, AT_EMPTY_PATH)
read(3, "# Locale name alias data base.\n#"..., 4096) = 2996
```

```
close(3)                                = 0
write(2, "mkdir: ", 7mkdir: )                   = 7
write(2, "cannot create directory \342\200\230foo2\342"..., 34cannot create directory ‘foo2’) = 34
openat(AT_FDCWD, "/usr/share/locale/en_GB/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file
openat(AT_FDCWD, "/usr/share/locale/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or
openat(AT_FDCWD, "/usr/share/locale-langpack/en_GB/LC_MESSAGES/libc.mo", O_RDONLY) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=3572, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 3572, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff0cdeae000
close(3)                                = 0
openat(AT_FDCWD, "/usr/share/locale-langpack/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No suc
write(2, ": File exists", 13: File exists)          = 13
```

Further Reading

- Understanding system calls on Linux with strace
  https://opensource.com/article/19/10/strace
- Linux strace Command Tutorial for Beginners
  https://www.howtoforge.com/linux-strace-command/

UNIVERSITY of York

File Protection and Access Control

Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

Outline

- Access Control
- Changing File Permissions
- Automating Tasks
- Further Reading

Types of Access and Access Control

Types of Access:

- **Read, Write, Execute, Append, Delete, List**

Access Control

- **Owner, Group, Universe**

```
-rw-rw-r--    1 pbg   staff      31200  Sep 3 08:30   intro.ps
drwx------    5 pbg   staff        512  Jul 8 09.33   private/
drwxrwxr-x    2 pbg   staff        512  Jul 8 09:35   doc/
drwxrwx---    2 jwg   student      512  Aug 3 14:13   student-proj/
-rw-r--r--    1 pbg   staff       9423  Feb 24 2012   program.c
-rwxr-xr-x    1 pbg   staff      20471  Feb 24 2012   program
drwx--x--x    4 tag   faculty      512  Jul 31 10:31  lib/
drwx------    3 pbg   staff       1024  Aug 29 06:52  mail/
drwxrwxrwx    3 pbg   staff        512  Jul 8 09:35   test/
```

Types of Access and Access Control

Read: Read from the file.

Write: Write or rewrite the file.

Execute: Load the file into memory and execute it.

Append: Write new information at the end of the file.

Delete: Delete the file and free its space for possible reuse.

List: List the name and attributes of the file.

File Permissions

File permissions control who can read, write, and execute files.

They are represented by a set of three categories of permissions: **owner**, **group**, and **universe**.

Each category has three types of permissions:

1. **Read (r)**: The ability to read the contents of the file.
2. **Write (w)**: The ability to modify or delete the file.
3. **Execute (x)**: The ability to execute the file if it's a program or script, or to access it if it's a directory.

Types of Access and Access Control

Systems recognize three classifications of users in connection with each file:

- **Owner:** The user who created the file is the owner.
- **Group:** A set of users who are sharing the file and need similar access is a group, or work group.
- **Universe:** All other users in the system constitute the universe.

Changing File Permissions

```
ub@ub:~$ echo 'echo "Hello, World!"' > ~/myscript.sh
ub@ub:~$ ./myscript.sh
bash: ./myscript.sh: Permission denied
ub@ub:~$ ls -l ./myscript.sh
-rw-rw-r-- 1 ub ub 21 Jan 23 16:01 ./myscript.sh
ub@ub:~$ chmod +x ~/myscript
chmod: cannot access '/home/ub/myscript': No such file or directory
ub@ub:~$ chmod +x ~/myscript.sh
ub@ub:~$ ls -l ./myscript.sh
-rwxrwxr-x 1 ub ub 21 Jan 23 16:01 ./myscript.sh
ub@ub:~$ ./myscript.sh
Hello, World!
ub@ub:~$
```

Changing File Permissions

```
ub@ub:~/foo$ touch output.txt
ub@ub:~/foo$ ls -l output.txt
-rw-rw-r-- 1 ub ub 0 Feb 28 15:46 output.txt
ub@ub:~/foo$ chmod 777 output.txt
ub@ub:~/foo$ ls -l output.txt
-rwxrwxrwx 1 ub ub 0 Feb 28 15:46 output.txt
ub@ub:~/foo$ chmod 444 output.txt
ub@ub:~/foo$ ls -l output.txt
-r--r--r-- 1 ub ub 0 Feb 28 15:46 output.txt
ub@ub:~/foo$ chmod 764 output.txt
ub@ub:~/foo$ ls -l output.txt
-rwxrw-r-- 1 ub ub 0 Feb 28 15:46 output.txt
ub@ub:~/foo$
```

Changing File Permissions

Open | *change_permissions.c ~/foo | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>

int main() {
    const char *filename = "output.txt";
    const char *text = "Hello World.\n";

    // Open the file for writing
    // read/write permissions for owner, group, and others
    int fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0666);

    // Write the text to the file
    write(fd, text, strlen(text));
    // Close the file
    close(fd);

    // Change file permissions
    // read-only for owner, group, and others
    chmod(filename, 0444);

    printf("File \"%s\" written and permissions set to read-only.\n", filename);
    return 0;
}
```

Changing File Permissions

```
ub@ub:~/foo$ gcc -o change_permissions change_permissions.c
ub@ub:~/foo$ ./change_permissions
File "output.txt" written and permissions set to read-only.
ub@ub:~/foo$ ls -l output.txt
-r--r--r-- 1 ub ub 13 Feb 28 15:40 output.txt
ub@ub:~/foo$ cat output.txt
Hello World.
ub@ub:~/foo$
```

```
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
write(3, "Hello World.\n", 13)          = 13
close(3)                                = 0
chmod("output.txt", 0444)               = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
getrandom("\xe0\xb8\x81\x95\x87\x8b\xde\x06", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5609e4b8c000
brk(0x5609e4bad000)                     = 0x5609e4bad000
write(1, "File \"output.txt\" written and pe"..., 60File "output.txt" written and permissions
 set to read-only.
```

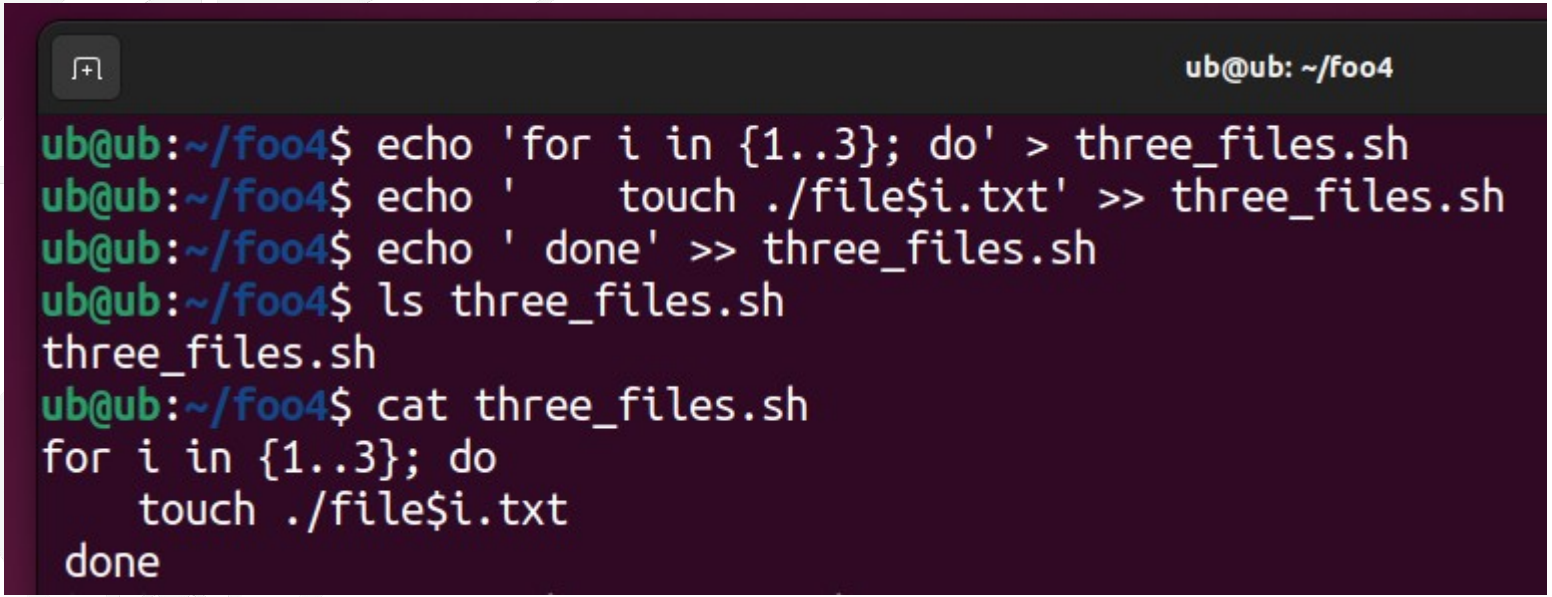
Adding a new user

```
ub@ub:~$ echo 'echo "Hello!' > my_script.sh
ub@ub:~$ ls -l my_script.sh
-rw-rw-r-- 1 ub ub 13 Mar  9 16:53 my_script.sh
ub@ub:~$ sudo adduser john
Adding user `john' ...
Adding new group `john' (1001) ...
Adding new user `john' (1001) with group `john' ...
The home directory `/home/john' already exists.  Not copying from `/etc/skel'.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Changing the user information for john
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

Changing file owner

```
ub@ub:~$ sudo chown john my_script.sh
ub@ub:~$ ls -l my_script.sh
-rw-rw-r-- 1 john ub 13 Mar  9 16:53 my_script.sh
ub@ub:~$ sudo userdel john
ub@ub:~$ ls -l my_script.sh
-rw-rw-r-- 1 1001 ub 13 Mar  9 16:53 my_script.sh
ub@ub:~$ sudo chown ub my_script.sh
ub@ub:~$ ls -l my_script.sh
-rw-rw-r-- 1 ub ub 13 Mar  9 16:53 my_script.sh
ub@ub:~$
```

Automating file creation



Automating file creation

Open | three_files.sh ~/foo4 | Save

```
1 for i in {1..3}; do
2     touch ./file$i.txt
3 done
```

sh | Tab Wi

```
ub@ub:~/foo4$ ls -l three_files.sh
-rw-rw-r-- 1 ub ub 49 Mar  9 17:07 three_files.sh
ub@ub:~/foo4$ ./three_files.sh
bash: ./three_files.sh: Permission denied
ub@ub:~/foo4$ chmod +x three_files.sh
ub@ub:~/foo4$ ls -l three_files.sh
-rwxrwxr-x 1 ub ub 49 Mar  9 17:07 three_files.sh
ub@ub:~/foo4$ ./three_files.sh
ub@ub:~/foo4$ ls
file1.txt  file2.txt  file3.txt  three_files.sh
ub@ub:~/foo4$
```

Further Reading

- Linux file permissions explained
  https://www.redhat.com/sysadmin/linux-file-permissions-explained
- Classic SysAdmin: Understanding Linux File Permissions
  https://www.linuxfoundation.org/blog/blog/classic-sysadmin-understanding-linux-file-permissions

Introduction to Malware

Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

What is Malware?

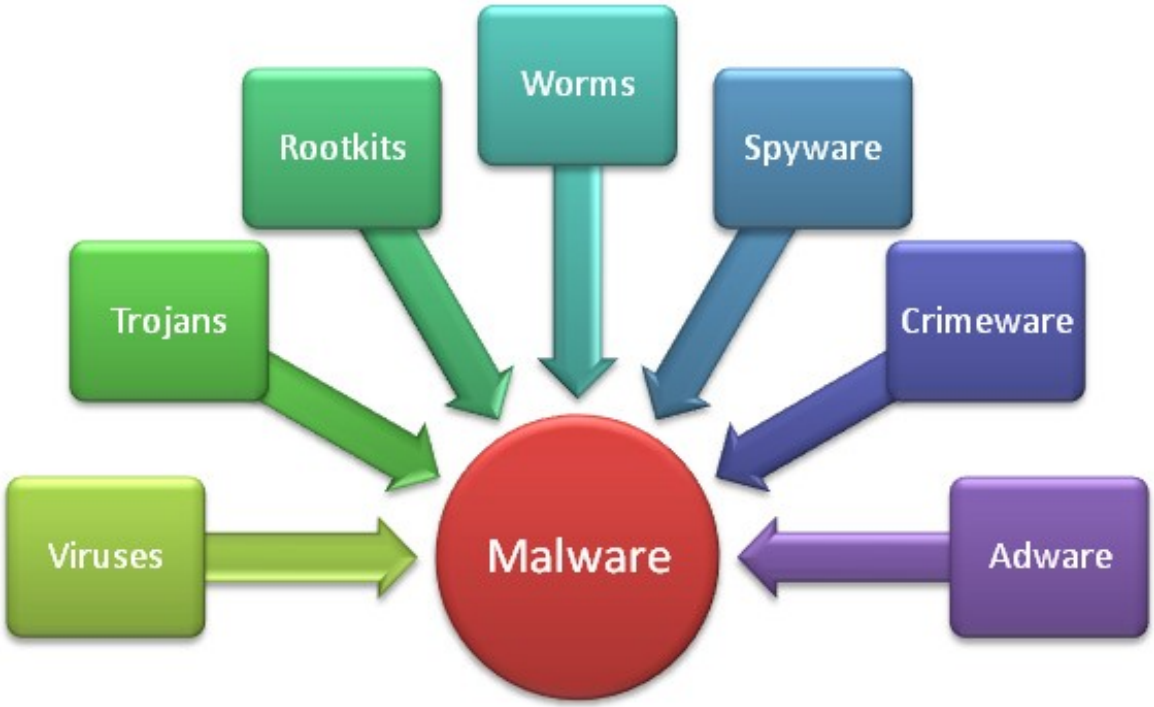
- **Malicious** software (program) that can be used to:
  - **compromise** computer functions
  - **steal** data
  - **bypass** access control
  - or otherwise cause **harm** to the computer.

Definition

NIST Special Publication (SP) 800-83 (Guide to Malware Incident Prevention and Handling for Desktops and Laptops, Revision 1, July 2013) defines malware as:

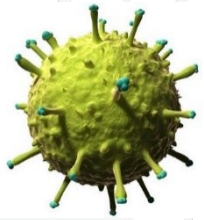
"a program that is **covertly** inserted into another program with the intent to **destroy** data, run destructive or **intrusive** programs, or otherwise compromise the **confidentiality**, **integrity**, or **availability** of the victim's **data**, **applications**, or **operating system**."

Common Types of Malware



Common Types of Malware

Virus



Worm

Trojan

Backdoor

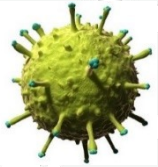
Spyware

Keylogger

Ransomware

Virus, Worm, Trojan

Virus



When executed, tries to replicate itself into **other executable code**

Worm

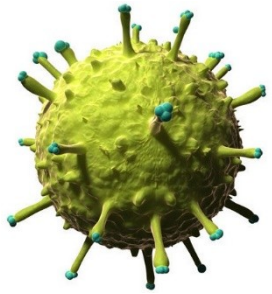
Can **run independently** and can **propagate** a complete working version of itself into other hosts on a network

Trojan

Appears to have a useful function, but also has a **hidden** and potentially malicious function

Virus

- Malware that, when executed, tries to replicate itself **into other executable code**
  - when it succeeds the code is said to be **infected**.
- Attaches itself to a **legitimate** program – when you launch the program, you **launch** the virus.
- Requires user **interaction** to infect the computer.



Worm

- A computer program that can **run independently** and can **propagate** a complete working version of itself into other hosts on a network.
- Unlike virus, a worm **does not infect** other programs and **does not require execution** to launch.
- Spreads via a computer **network** by exploiting operating system vulnerabilities.

Trojan horse (1/2)

- A computer program that appears to have a useful function, but also has a **hidden** and potentially malicious function
  - Presents itself as **harmless** to **disguise** a **malicious** effect.
  - Does something the user **does not expect.**

Trojan horse (2/2)

- Sometimes evades security mechanisms by exploiting **legitimate authorizations** of a system that invokes the Trojan horse program.
- **Does not replicate**
- A Trojan can give a malicious party **remote access** to an infected computer

Types of Trojans

- Continuing to perform the function of the original program and **additionally** performing a separate malicious activity

- Continuing to perform the function of the original program but **modifying** this function to perform malicious activity or to disguise other malicious activity

- Performing a malicious function that **completely replaces** the function of the original program

Spyware, Keylogger

Spyware

Collects and sends information about your computer usage to a **third party.**

Keylogger

Captures key strokes (**key logging)** and/or **tracks cookies.**

Spyware (1/2)

- **Collects** and sends information about your computer usage to a **third party.**
- **Covertly** runs in the background without the user's knowledge.
- Commonly **delivered** via a **Trojan Horse.**

Spyware (2/2)

- Captures key strokes (**key logging)** and/or **tracks cookies.**

- Computers **slow down** as a consequence.

Ransomware, Scareware

Ransomware

Encrypts your data and only decrypts it after a **ransom** is paid.

Scareware

Uses **scare tactics** to obtain money or information from a user

Ransomware

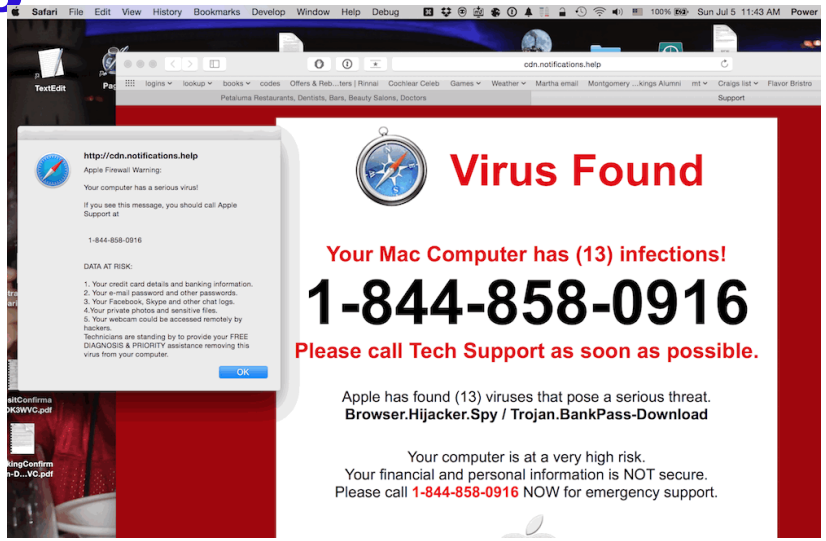
- **Encrypts** your data and only decrypts it after a **ransom** is paid.

Scareware

- Uses **scare tactics** to obtain money or information from a user.

- Usually acts as a **virus checker** which tries to sell a **(fake) virus remover.**

Scareware examples



Other Types of Malware

Name	Description
Exploit	Code specific to a single vulnerability or set of vulnerabilities.
Downloader	Program that installs other items on a machine that is under attack. Usually a downloader is sent in an email.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Flooder	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Rootkit	Set of hacker tools used after hacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Mobile code	Software (e.g., scripts, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.

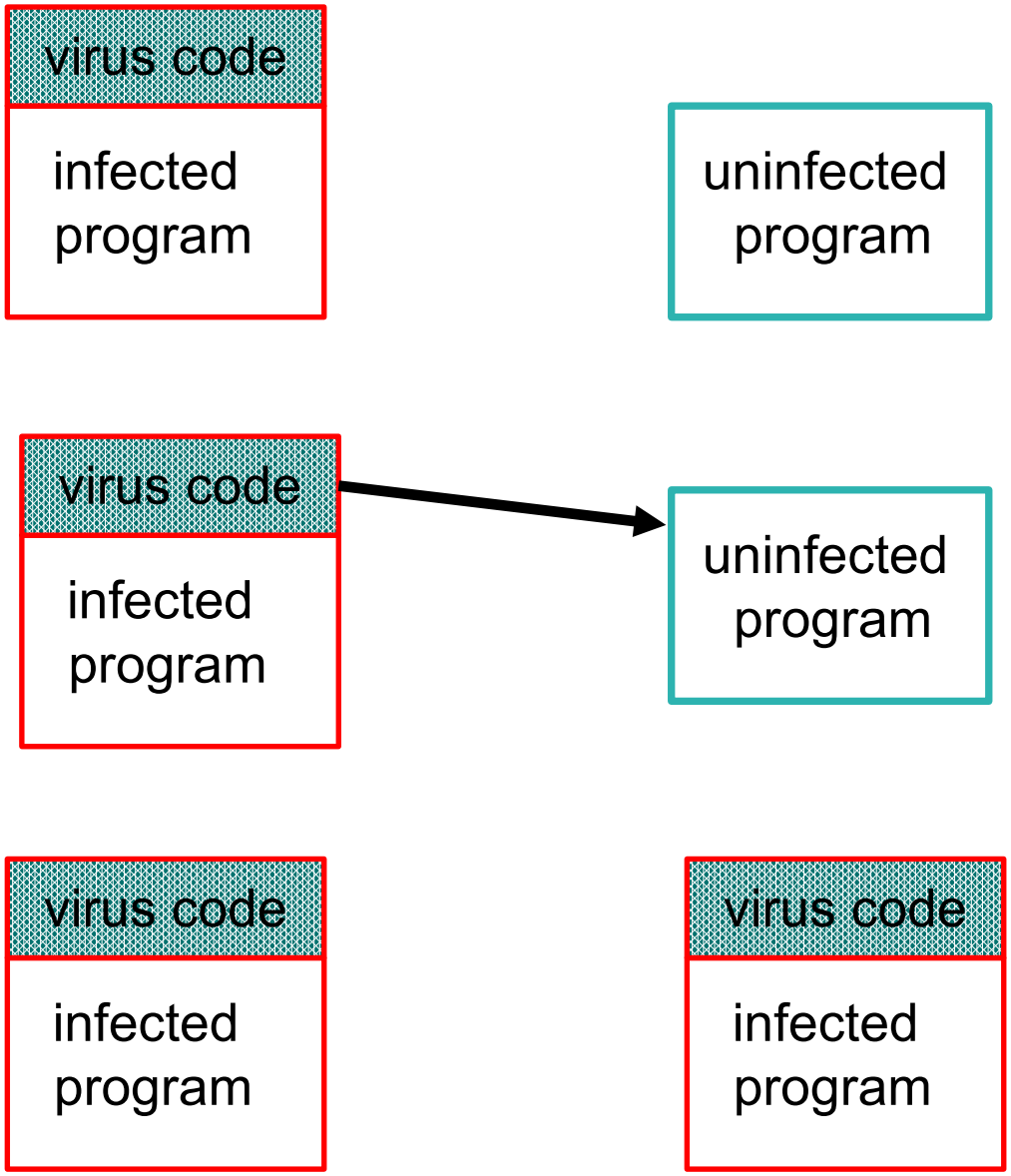
UNIVERSITY of York

Simple Examples of Malware

Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

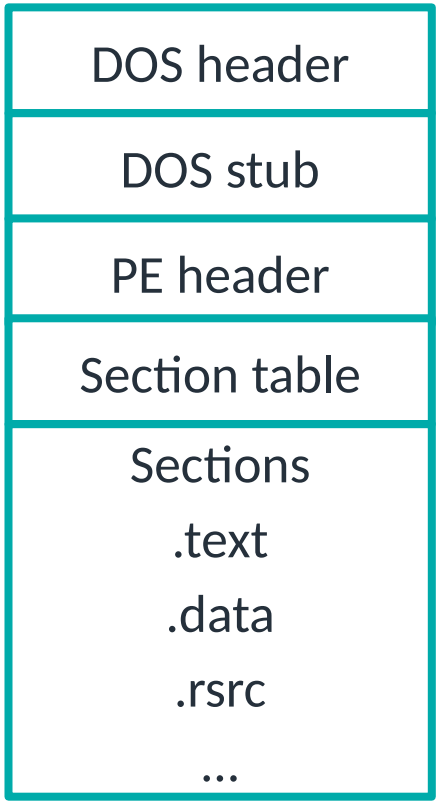
Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

Infecting with a virus



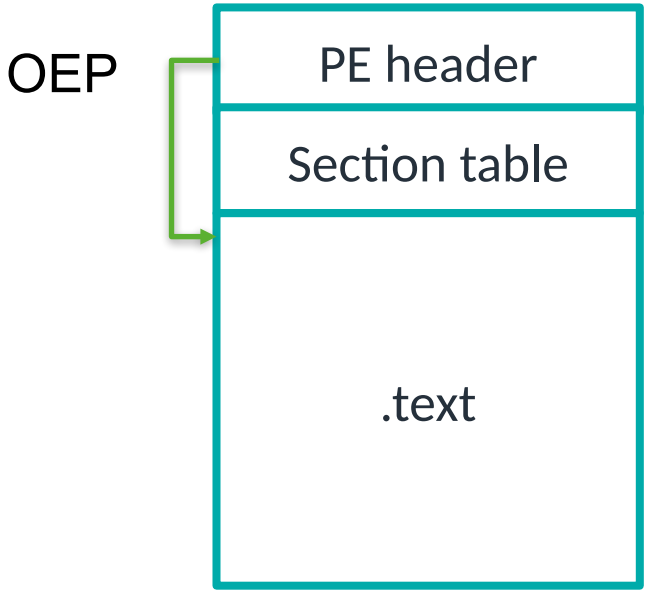
PE File Virus

- Portable Executable (PE) File
  - Windows 32-bit and 64-bit OS
  - Executable files (.exe), Dynamic-link libraries (.dll)



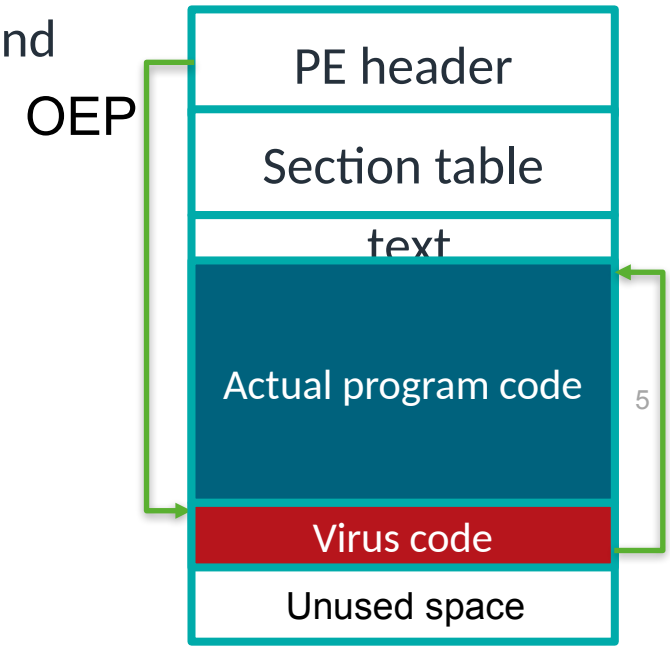
PE File Virus

- .text section: contains program code
- .data section: global variables
- Original Entry Point (OEP)



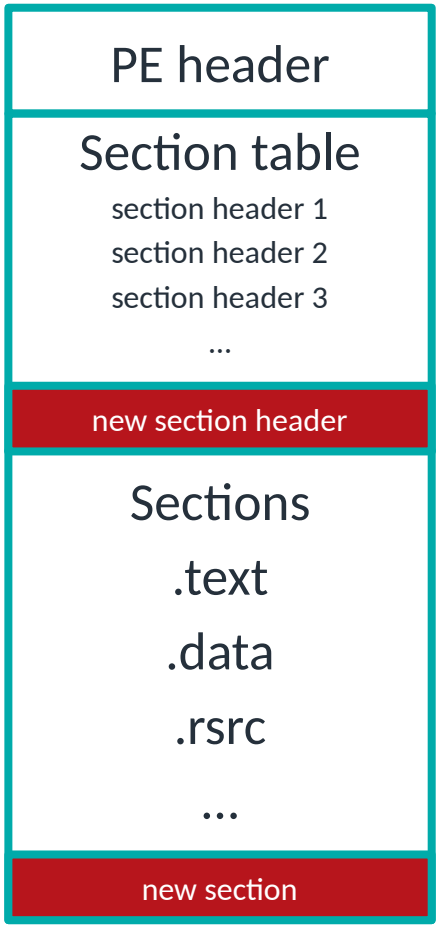
PE File Virus

- **Infecting unused .text section**
  - Locating the beginning of unused space of .text
  - Appending virus code
  - Replacing OEP with beginning address of virus code
  - Returning to the actual program code in the end



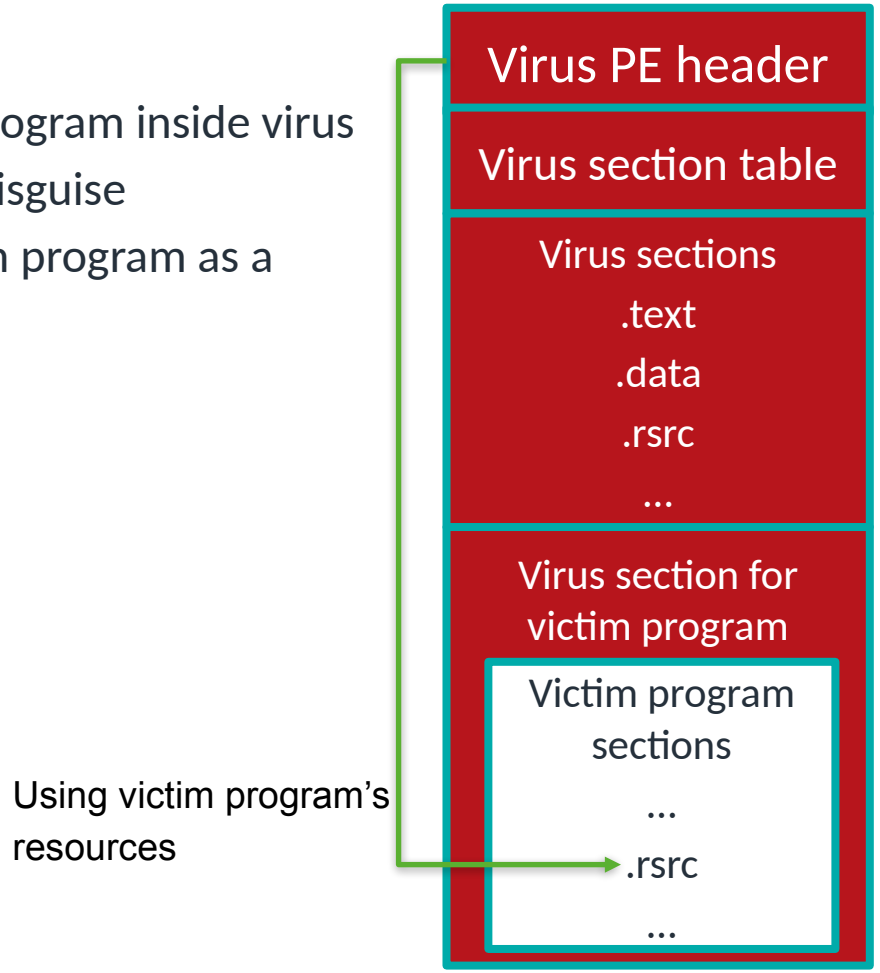
PE File Virus

- **Appending new sections**
  - Appending a new section header in the section table
  - Appending a new section after existing sections



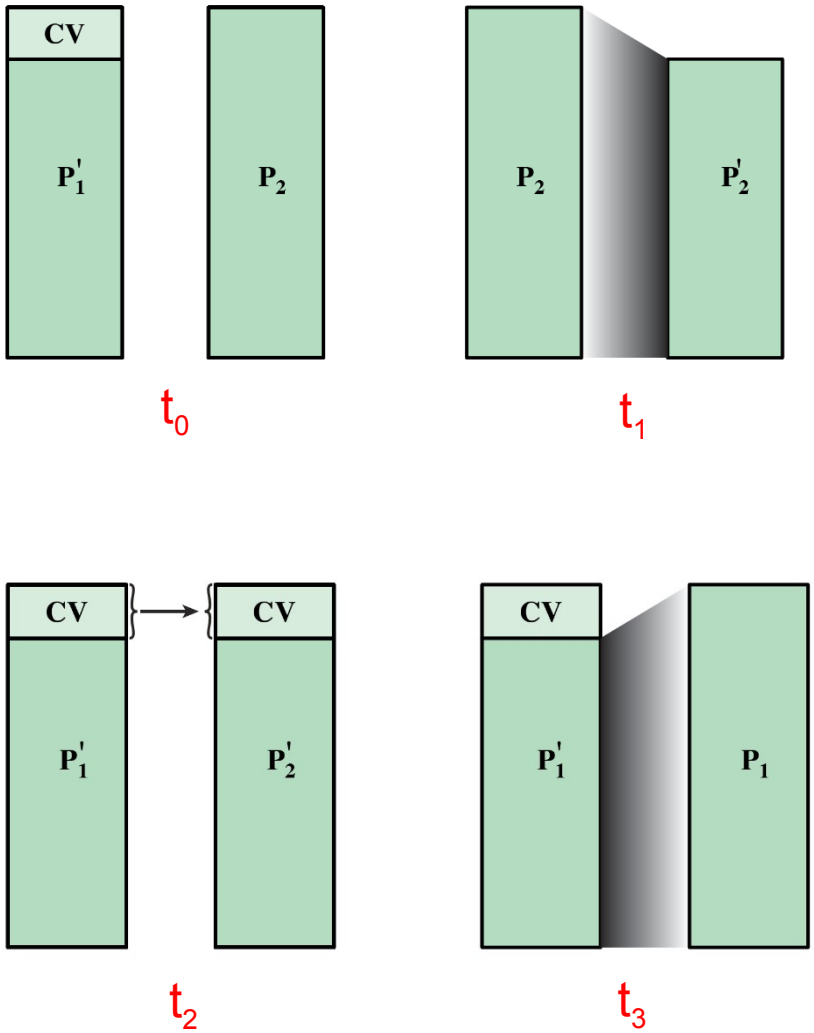
PE File Virus

- **Compression Virus**
  - Compressing and appending victim program inside virus
  - Using victim program's resources to disguise
  - When running, uncompress the victim program as a tempfile and run it

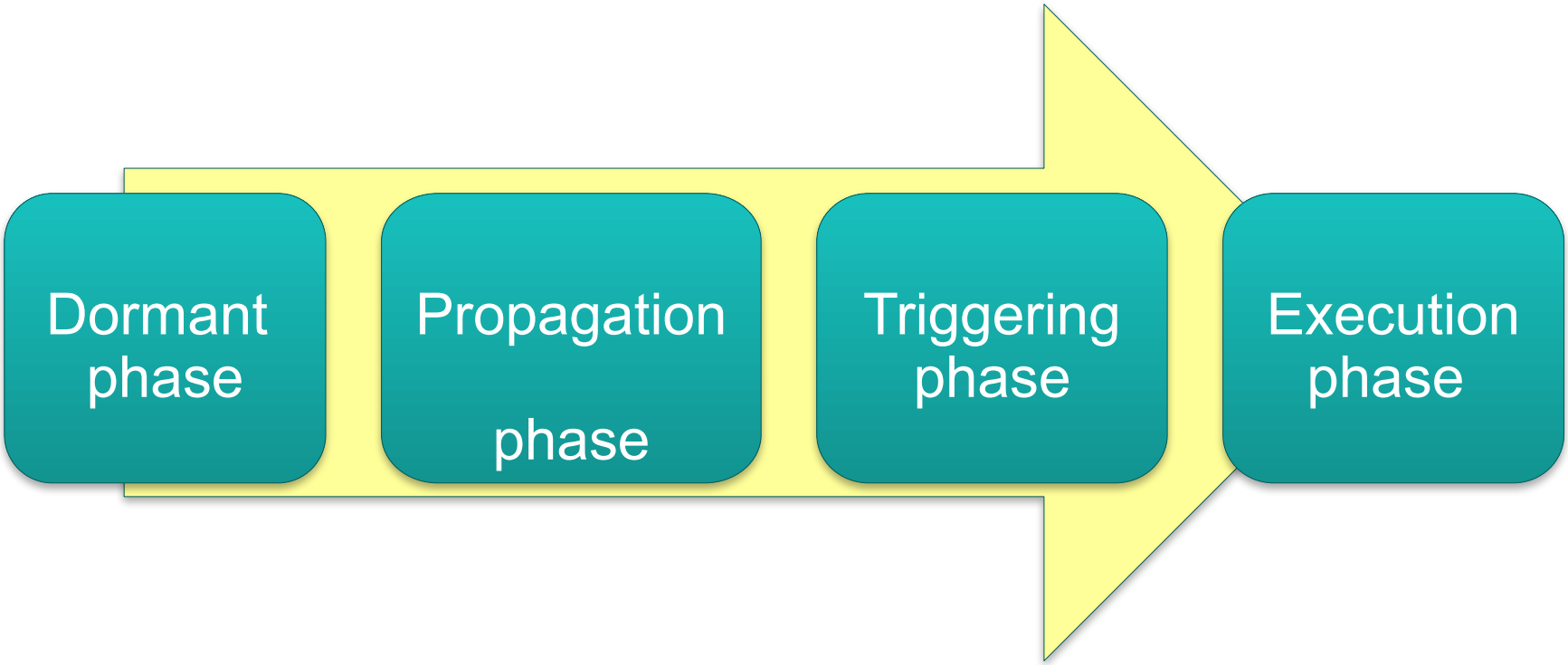


Compression Virus

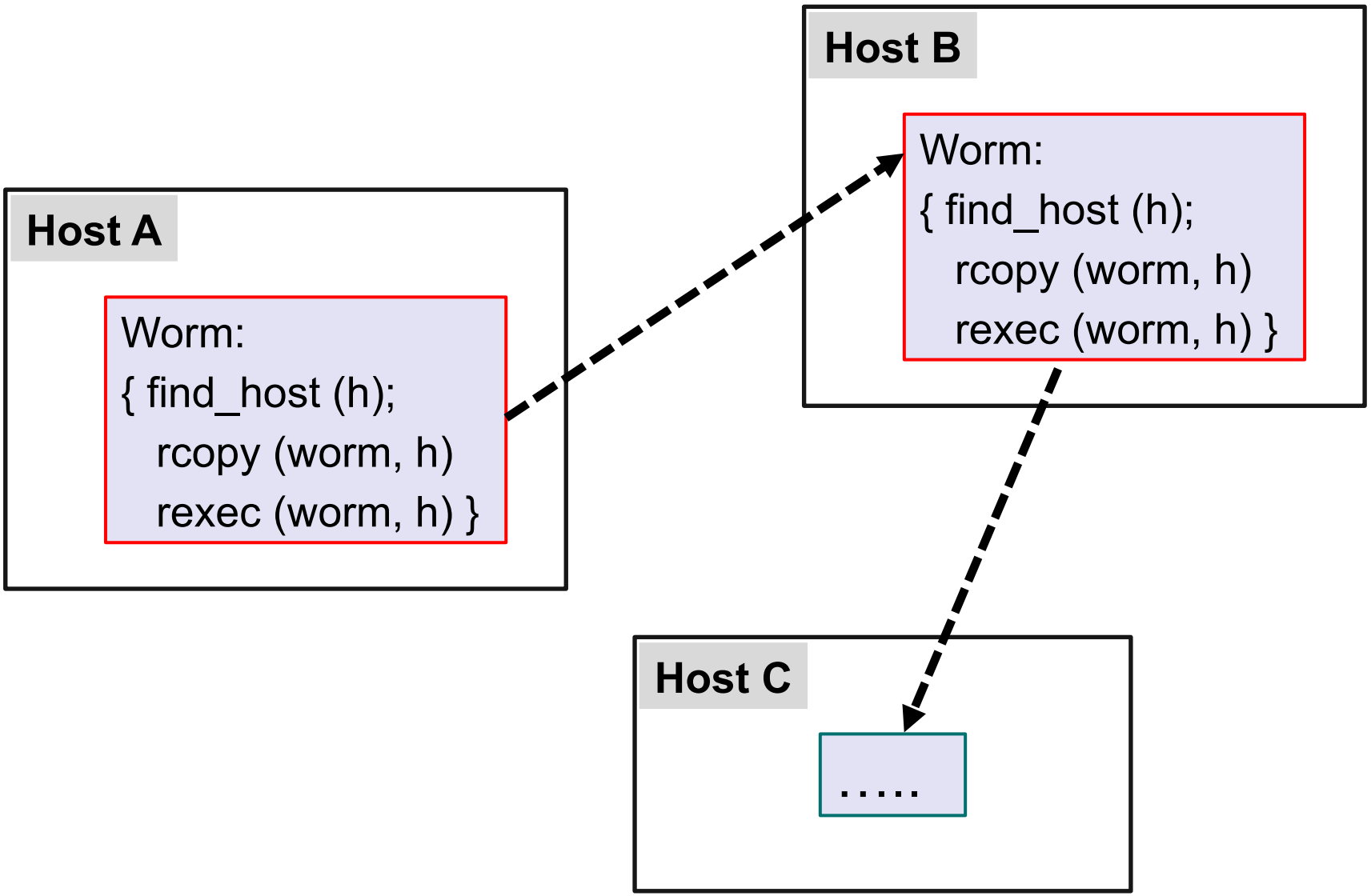
```
program CV
1234567;

procedure attach-to-program;
begin
   repeat
      file := get-random-program; t0
   until first-program-line ≠ 1234567;
   compress file; t1
   prepend CV to file; t2
end;

begin (* main action block *)
   if ask-permission then attach-to-program;
   uncompress rest of this file into tempfile; t3
   execute tempfile; t4
end;
```



Phases of a Virus

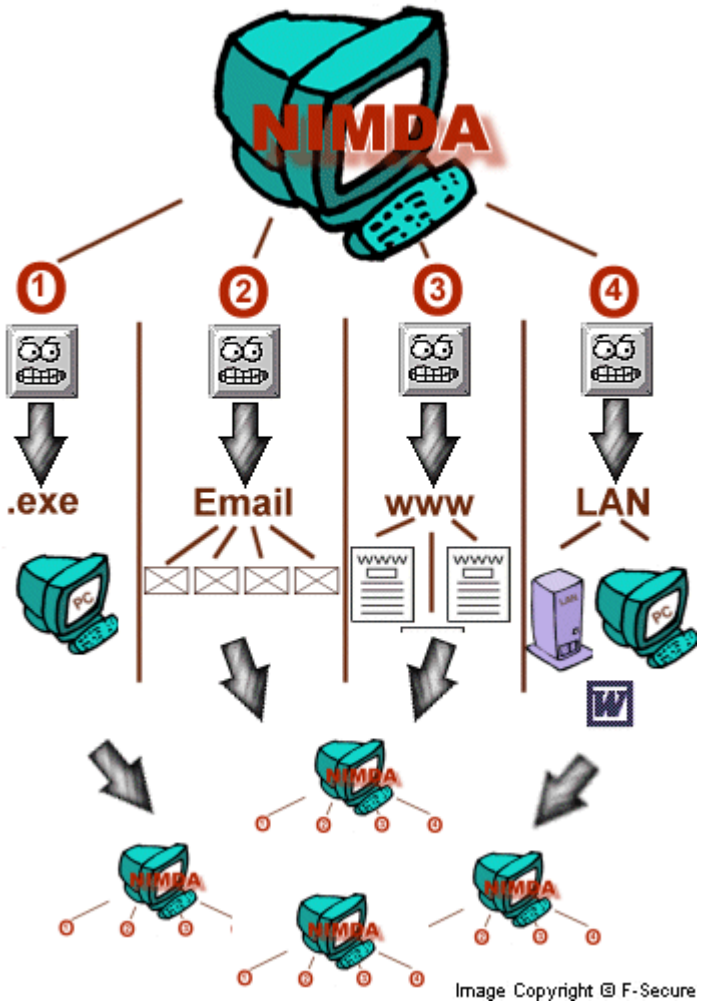


A simple worm



Net-worm:W32/Nimda

- 18 Sep 2001
- Windows 95, 98, NT, 2000, XP, and servers
- File infection (Virus behaviour)
- Emails (containing README.EXE)
- Web pages (web server vulnerabilities)
- LAN (place RICHED20.DLL through LAN)



Logic Bomb

- Code embedded in the malware that is set to **“explode”**
- Lies dormant until a **predefined condition** is met
  - the program then **triggers** an unauthorized act

```
legitimate code
if  (date is Friday the 13th)
    crash_computer();
legitimate code
```

What conditions could be used as triggers?

Conditions

Examples of **conditions** that can be used as triggers:

- presence or absence of certain **files** or **devices** on the system
- a particular day of the week or **date**
- a particular **version** or **configuration** of some software
- a particular **user** running the application

Once triggered, a bomb may:

- alter or **delete** data or entire files
- cause a machine **halt**
- do some other **damage**

Backdoor (Trapdoor)

- Any mechanism that **bypasses** a normal security check.
- A secret entry point into a program that may allow **unauthorized access** to functionality or data.

Backdoor (Trapdoor)

Normal login program sketch:

```
login
        …
        print “Type password:”
        read (password)
        if valid (password)
                then permit
        else deny
end
```

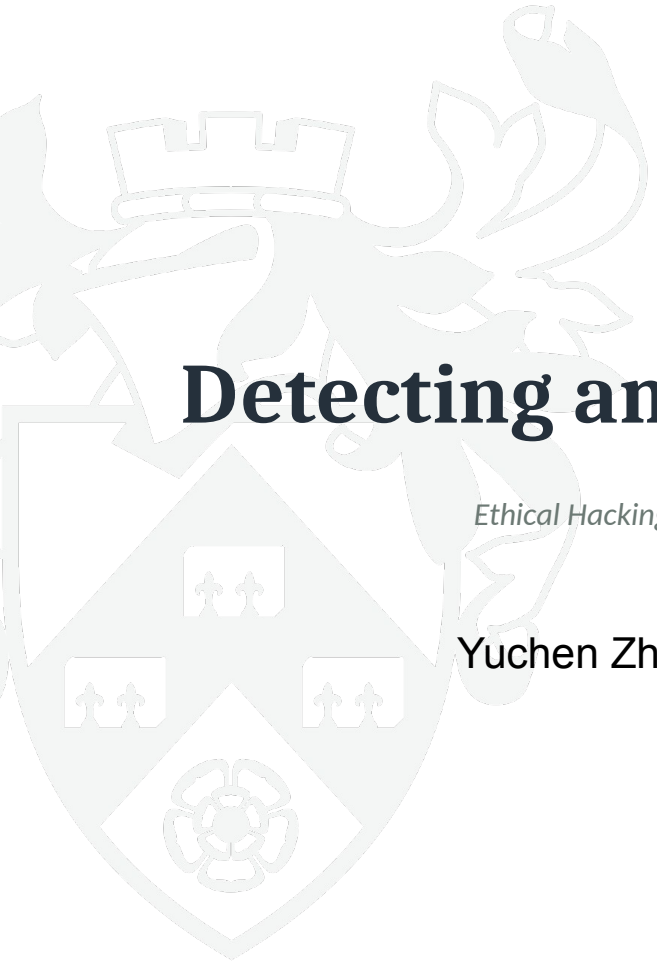
How to put a backdoor here?

Backdoor (Trapdoor)

"Backdoor" login program sketch:

```
login
        …
        print "Type password:"
        read (password)
        if valid (password) or (password="abc123")
                then permit
        else deny
end
```

The password *abc123* is an unknown functionality to the user (i.e., does not approach in the manual), but can be used as trap door by someone who knows it



Detecting and Exploiting Vulnerabilities

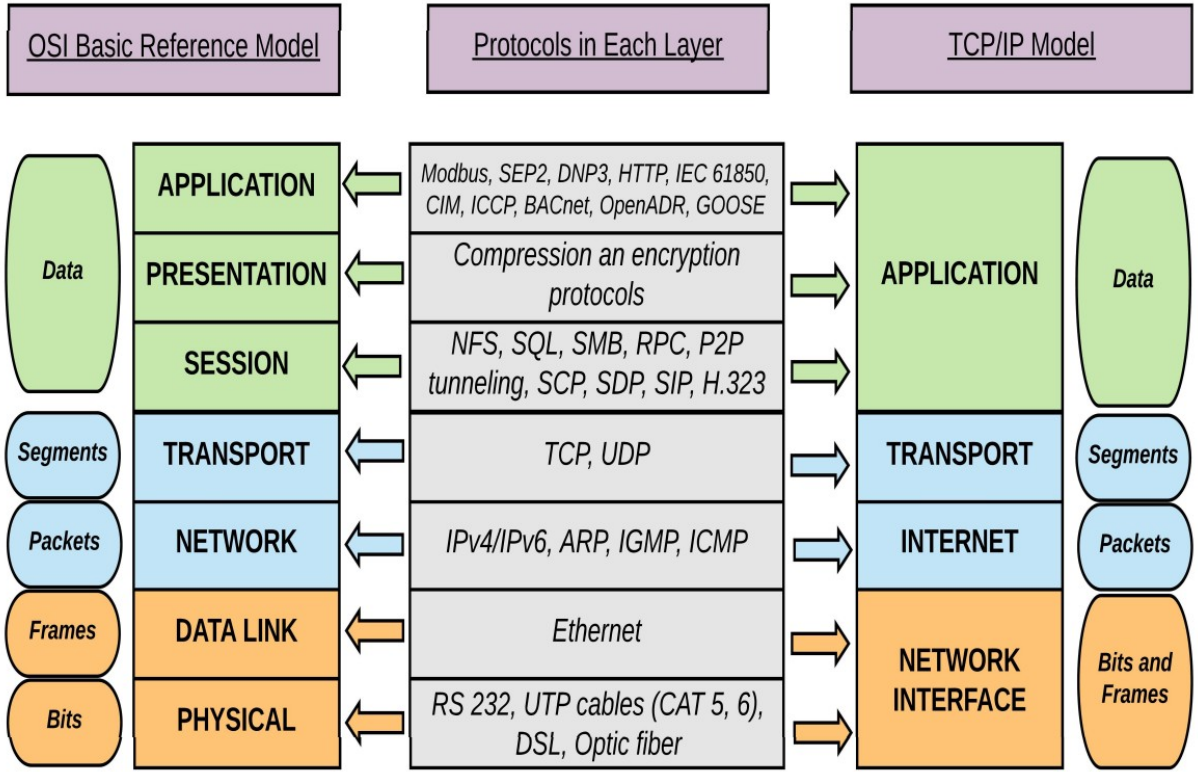
Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

Outline

- Basics of Networks
  - OSI and TCP/IP models
  - Capturing packets
  - Testing reachability of a host
  - Measuring network performance
- Network Scanning
  - Nmap
  - NetCat
- Exploiting Vulnerabilities
  - Directory traversal
- Supplementary Reading

Basics of Networks

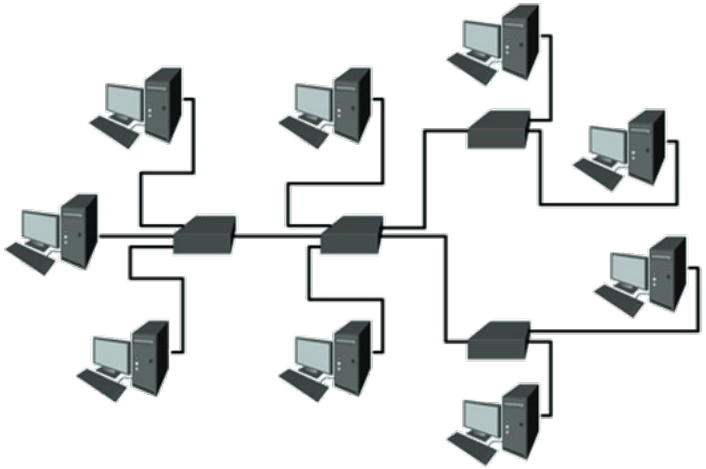


Address Resolution Protocol (ARP)

1. Request: looking for MAC address of IP 10.0.0.1

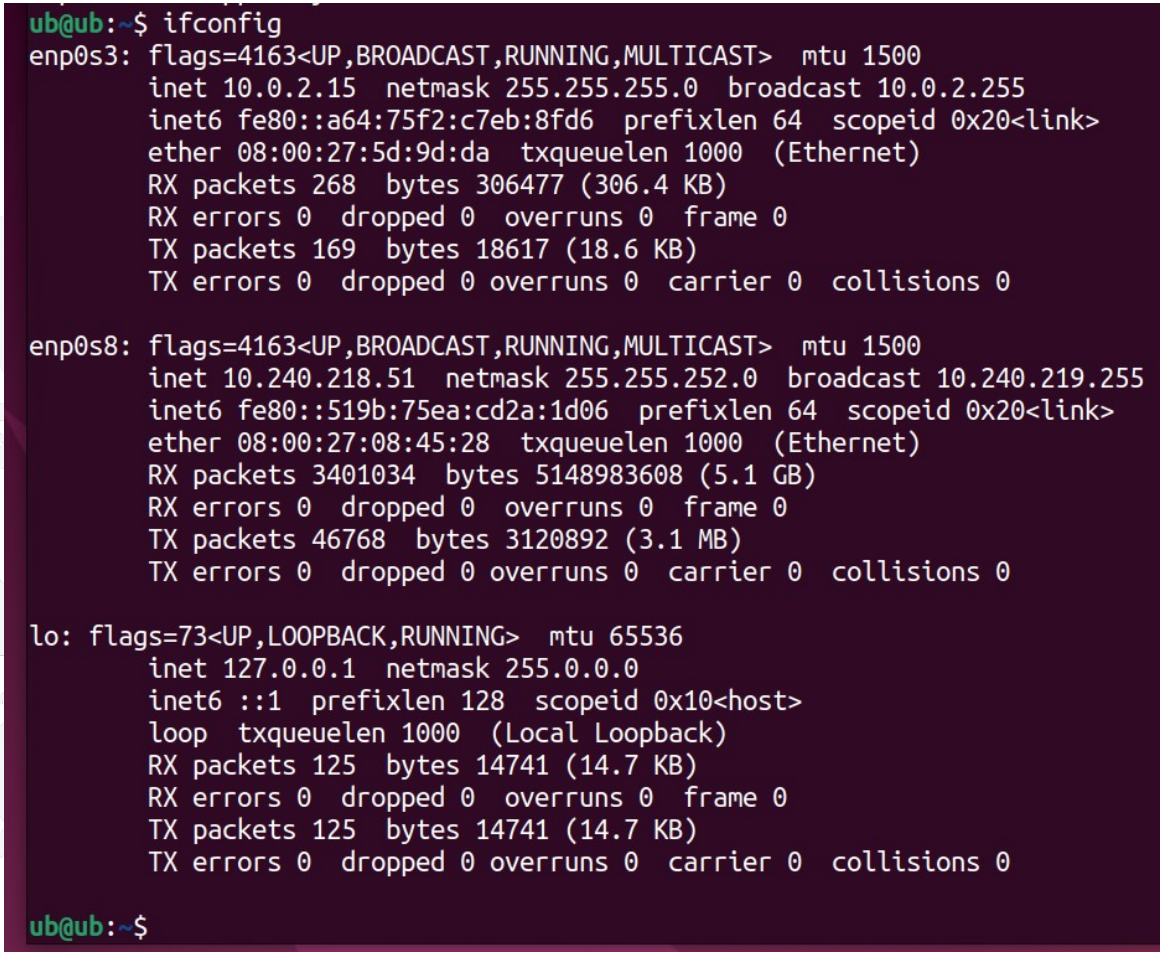
2. Response (from 10.0.0.1): My MAC address is …

3. Cache



Capturing network traffic: Tcpdump

```
ub@ub:~$ sudo tcpdump -i enp0s8
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:15:50.625496 IP6 fe80::e393:da59:a8c0:fe89 > ip6-allrouters: ICMP6, router solicitation, length 8
18:15:52.854979 ARP, Request who-has wifistr2-3812.york.ac.uk tell wifistr2-3812.york.ac.uk, length 46
18:15:56.134460 ARP, Request who-has wifistr2-3854.york.ac.uk tell wifistr2-3854.york.ac.uk, length 46
```


ub@ub:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
inet6 fe80::a64:75f2:c7eb:8fd6 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:5d:9d:da txqueuelen 1000 (Ethernet)
RX packets 268 bytes 306477 (306.4 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 169 bytes 18617 (18.6 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.240.218.51 netmask 255.255.252.0 broadcast 10.240.219.255
inet6 fe80::519b:75ea:cd2a:1d06 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:08:45:28 txqueuelen 1000 (Ethernet)
RX packets 3401034 bytes 5148983608 (5.1 GB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 46768 bytes 3120892 (3.1 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 125 bytes 14741 (14.7 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 125 bytes 14741 (14.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
ub@ub:~$

Testing Reachability of a Host

```
ub@ub:~$ ping -c 5 -s 100 cisco.com
PING cisco.com (72.163.4.185) 100(128) bytes of data.
108 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=1 ttl=235 time=114 ms
108 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=2 ttl=235 time=114 ms
108 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=3 ttl=235 time=115 ms
108 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=4 ttl=235 time=114 ms
108 bytes from redirect-ns.cisco.com (72.163.4.185): icmp_seq=5 ttl=235 time=115 ms

--- cisco.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 113.649/114.392/115.316/0.561 ms
ub@ub:~$
```

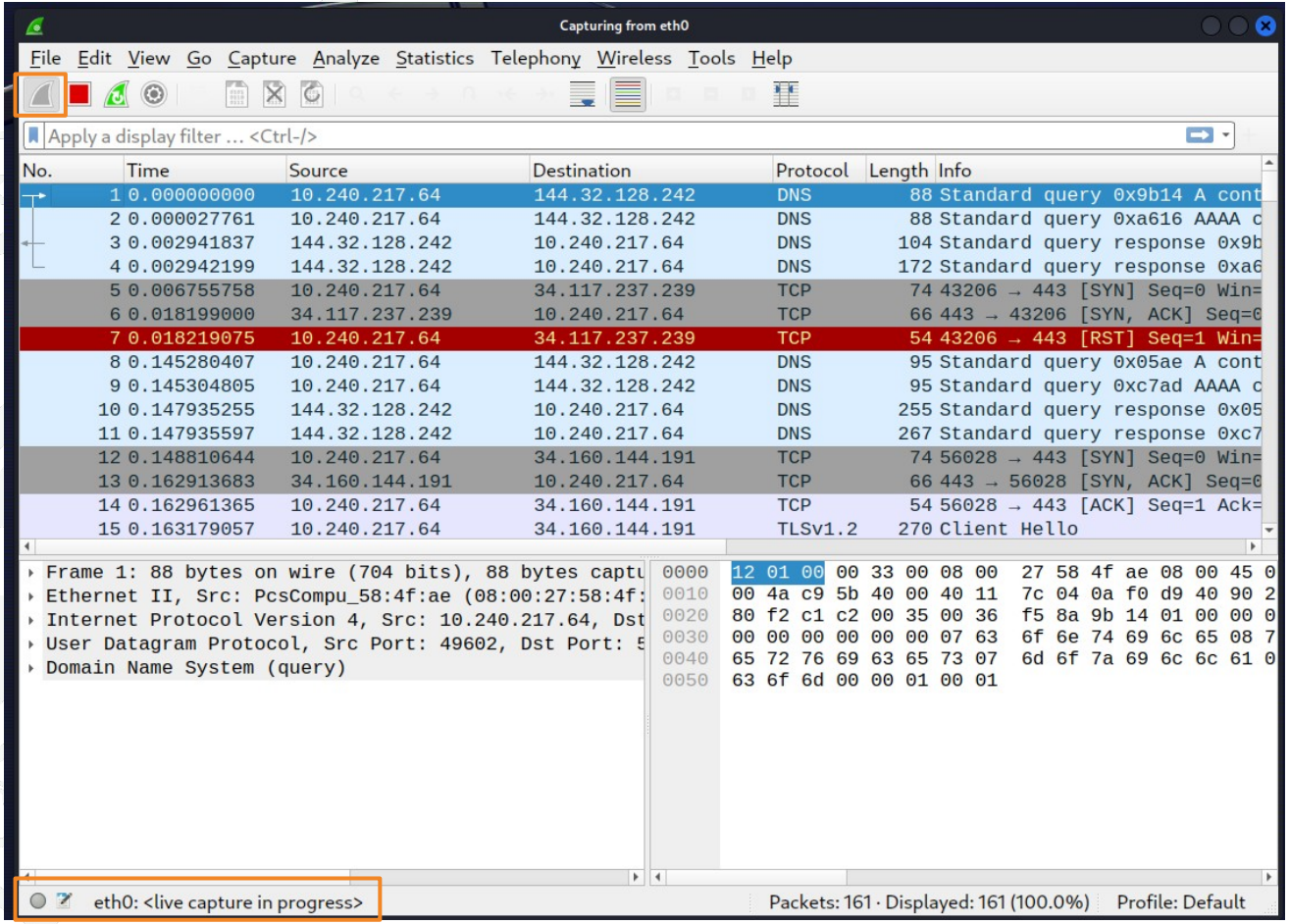
Tracing the Route to a Host

```
┌──(ka㉿kali)-[~]
└─$ traceroute google.com
traceroute to google.com (216.58.204.78), 30 hops max, 60 byte packets
 1  wifistr2-3854.york.ac.uk (10.240.216.1)  3.546 ms  3.478 ms  3.448 ms
 2  bsdccore2-2777.york.ac.uk (10.16.35.2)  3.413 ms  3.383 ms  3.351 ms
 3  sentry-659.york.ac.uk (144.32.65.69)  3.236 ms  3.153 ms  3.120 ms
 4  janetrouter1.york.ac.uk (144.32.255.226)  3.598 ms  3.571 ms  3.423 ms
 5  ae31-391.yorktd-rbr1.ja.net (146.97.150.89)  3.674 ms  3.633 ms  3.604 ms
 6  ae6.leedaq-rbr1.ja.net (146.97.71.149)  5.075 ms  6.595 ms  6.559 ms
 7  ae0.leedlu-rbr1.ja.net (146.97.71.125)  6.536 ms  6.519 ms  6.301 ms
 8  ae26.manckh-sbr2.ja.net (146.97.36.221)  7.259 ms  8.170 ms  8.135 ms
 9  ae29.erdiss-sbr2.ja.net (146.97.33.41)  9.439 ms  9.335 ms  9.309 ms
10  ae31.londpg-sbr2.ja.net (146.97.33.21)  12.289 ms  12.272 ms  12.256 ms
11  ae29.londhx-sbr1.ja.net (146.97.33.1)  13.346 ms  20.289 ms  14.697 ms
12  193.62.157.22 (193.62.157.22)  14.751 ms  16.466 ms  16.318 ms
13  * * *
14  209.85.241.92 (209.85.241.92)  14.154 ms 209.85.241.94 (209.85.241.94)  12
15  192.178.97.170 (192.178.97.170)  13.527 ms 192.178.46.87 (192.178.46.87)  1
16  142.251.232.211 (142.251.232.211)  14.787 ms lhr25s13-in-f14.1e100.net (216
 13.733 ms

┌──(ka㉿kali)-[~]
└─$
```

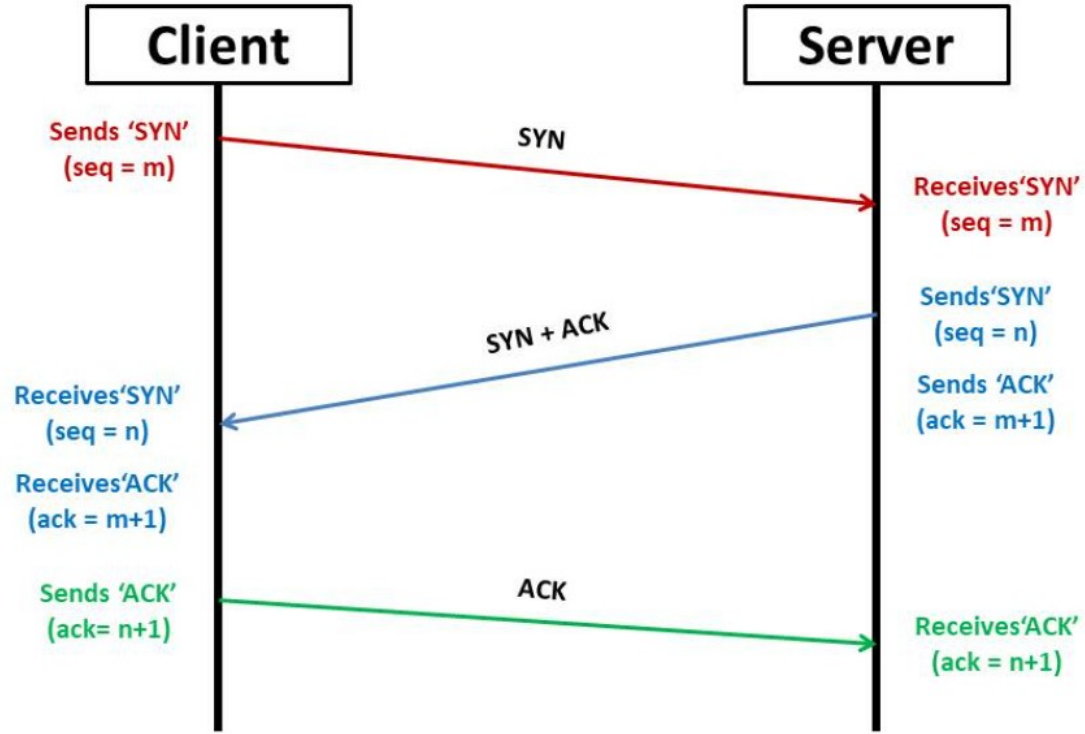
Capturing ICMP packets: Tcpdump

```
ub@ub:~$ ping -c4 -s 100 -i 2 10.240.218.51
PING 10.240.218.51 (10.240.218.51) 100(128) bytes of data.
108 bytes from 10.240.218.51: icmp_seq=1 ttl=64 time=0.577 ms
108 bytes from 10.240.218.51: icmp_seq=2 ttl=64 time=0.997 ms
108 bytes from 10.240.218.51: icmp_seq=3 ttl=64 time=0.891 ms
108 bytes from 10.240.218.51: icmp_seq=4 ttl=64 time=0.788 ms

--- 10.240.218.51 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 6011ms
rtt min/avg/max/mdev = 0.577/0.813/0.997/0.155 ms
ub@ub:~$
```

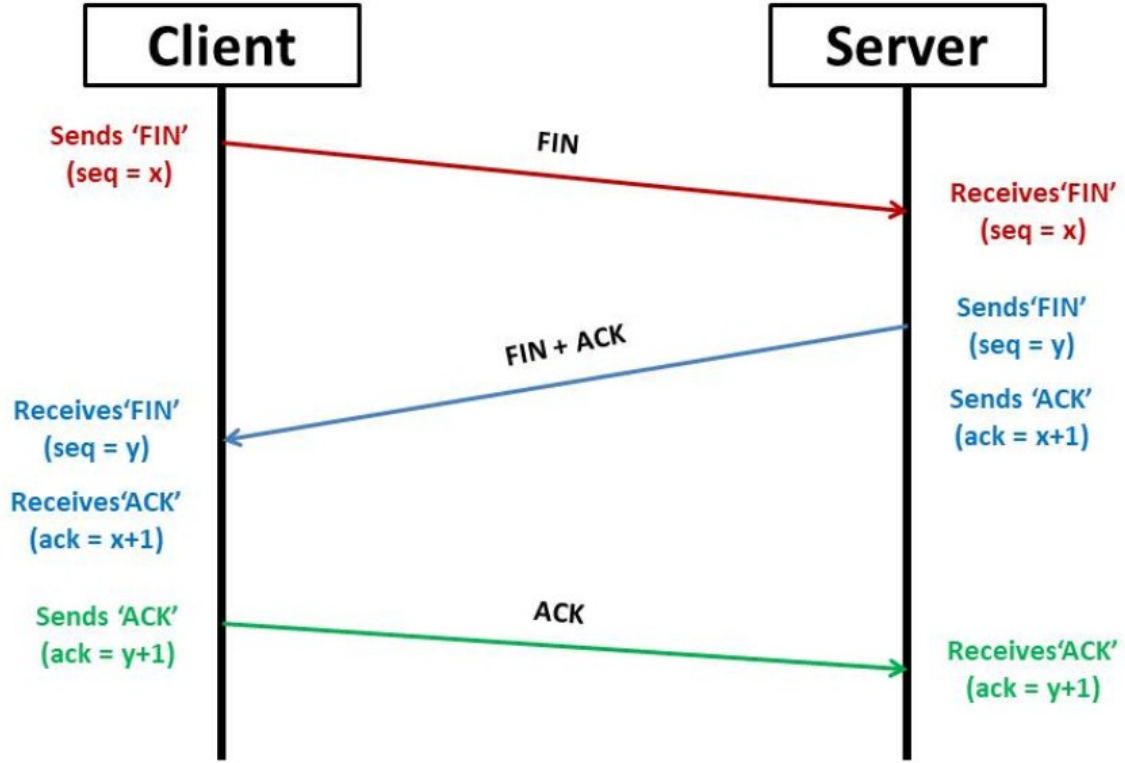
```
ub@ub:~$ sudo tcpdump -i enp0s8
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), snapshot length 262144 bytes
20:08:25.324856 IP nas-10-240-216-172.york.ac.uk > nas-10-240-218-51.york.ac.uk: ICMP echo request, id 5, seq 1, length 108
20:08:25.324878 IP nas-10-240-218-51.york.ac.uk > nas-10-240-216-172.york.ac.uk: ICMP echo reply, id 5, seq 1, length 108
20:08:25.379756 IP nas-10-240-218-51.york.ac.uk.34712 > ofns0.york.ac.uk.domain: 2224+ [1au] PTR? 51.218.240.10.in-addr.arpa. (55)
20:08:25.382484 IP ofns0.york.ac.uk.domain > nas-10-240-218-51.york.ac.uk.34712: 2224* 1/0/1 PTR nas-10-240-218-51.york.ac.uk. (97)
20:08:27.335921 IP nas-10-240-216-172.york.ac.uk > nas-10-240-218-51.york.ac.uk: ICMP echo request, id 5, seq 2, length 108
20:08:27.335956 IP nas-10-240-218-51.york.ac.uk > nas-10-240-216-172.york.ac.uk: ICMP echo reply, id 5, seq 2, length 108
20:08:29.336835 IP nas-10-240-216-172.york.ac.uk > nas-10-240-218-51.york.ac.uk: ICMP echo request, id 5, seq 3, length 108
20:08:29.336868 IP nas-10-240-218-51.york.ac.uk > nas-10-240-216-172.york.ac.uk: ICMP echo reply, id 5, seq 3, length 108
```


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 10.240.217.64 144.32.128.242 DNS 88 Standard query 0x9b14 A cont
2 0.000027761 10.240.217.64 144.32.128.242 DNS 88 Standard query 0xa616 AAAA c
3 0.002941837 144.32.128.242 10.240.217.64 DNS 104 Standard query response 0x9b
4 0.002942199 144.32.128.242 10.240.217.64 DNS 172 Standard query response 0xa6
5 0.006755758 10.240.217.64 34.117.237.239 TCP 74 43206 → 443 [SYN] Seq=0 Win=
6 0.018199000 34.117.237.239 10.240.217.64 TCP 66 443 → 43206 [SYN, ACK] Seq=0
7 0.018219075 10.240.217.64 34.117.237.239 TCP 54 43206 → 443 [RST] Seq=1 Win=
8 0.145280407 10.240.217.64 144.32.128.242 DNS 95 Standard query 0x05ae A cont
9 0.145304805 10.240.217.64 144.32.128.242 DNS 95 Standard query 0xc7ad AAAA c
10 0.147935255 144.32.128.242 10.240.217.64 DNS 255 Standard query response 0x05
11 0.147935597 144.32.128.242 10.240.217.64 DNS 267 Standard query response 0xc7
12 0.148810644 10.240.217.64 34.160.144.191 TCP 74 56028 → 443 [SYN] Seq=0 Win=
13 0.162913683 34.160.144.191 10.240.217.64 TCP 66 443 → 56028 [SYN, ACK] Seq=0
14 0.162961365 10.240.217.64 34.160.144.191 TCP 54 56028 → 443 [ACK] Seq=1 Ack=
15 0.163179057 10.240.217.64 34.160.144.191 TLSv1.2 270 Client Hello
Frame 1: 88 bytes on wire (704 bits), 88 bytes captu
Ethernet II, Src: PcsCompu_58:4f:ae (08:00:27:58:4f:
Internet Protocol Version 4, Src: 10.240.217.64, Dst
User Datagram Protocol, Src Port: 49602, Dst Port: 5
Domain Name System (query)
0000 12 01 00 00 33 00 08 00 27 58 4f ae 08 00 45 0
0010 00 4a c9 5b 40 00 40 11 7c 04 0a f0 d9 40 90 2
0020 80 f2 c1 c2 00 35 00 36 f5 8a 9b 14 01 00 00 0
0030 00 00 00 00 00 00 07 63 6f 6e 74 69 6c 65 08 7
0040 65 72 76 69 63 65 73 07 6d 6f 7a 69 6c 6c 61 0
0050 63 6f 6d 00 00 01 00 01
eth0: <live capture in progress>
Packets: 161 · Displayed: 161 (100.0%)
Profile: Default

TCP: Connection Establishment



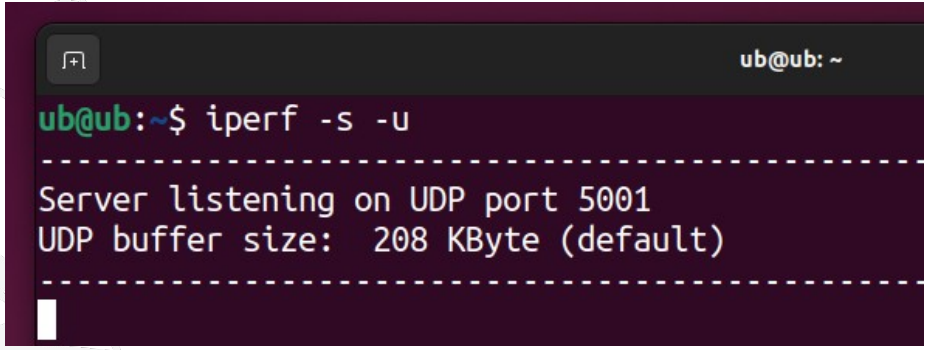
TCP: Connection Termination



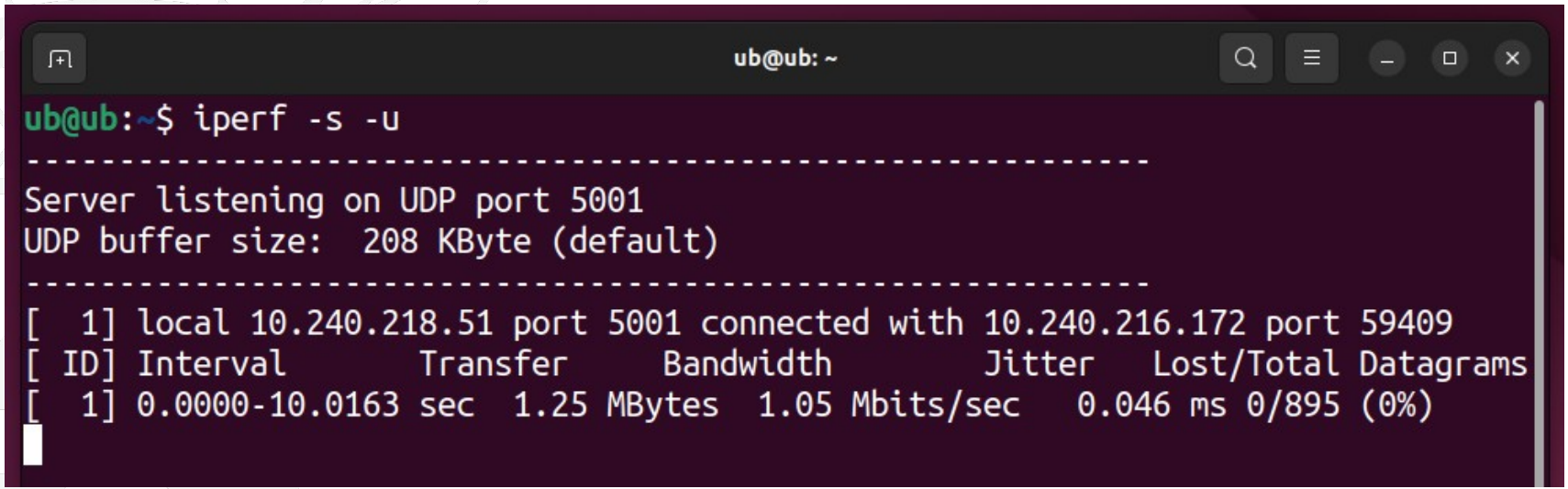
Measuring Network Performance

```
ub@ub:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 10.240.218.51 port 5001 connected with 10.240.216.172 port 58034
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0076 sec  4.87 GBytes  4.18 Gbits/sec
```

```
ub@ub:~$ iperf -c 10.240.218.51
------------------------------------------------------------
Client connecting to 10.240.218.51, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  1] local 10.240.216.172 port 58034 connected with 10.240.218.51 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0072 sec  4.87 GBytes  4.18 Gbits/sec
ub@ub:~$
```

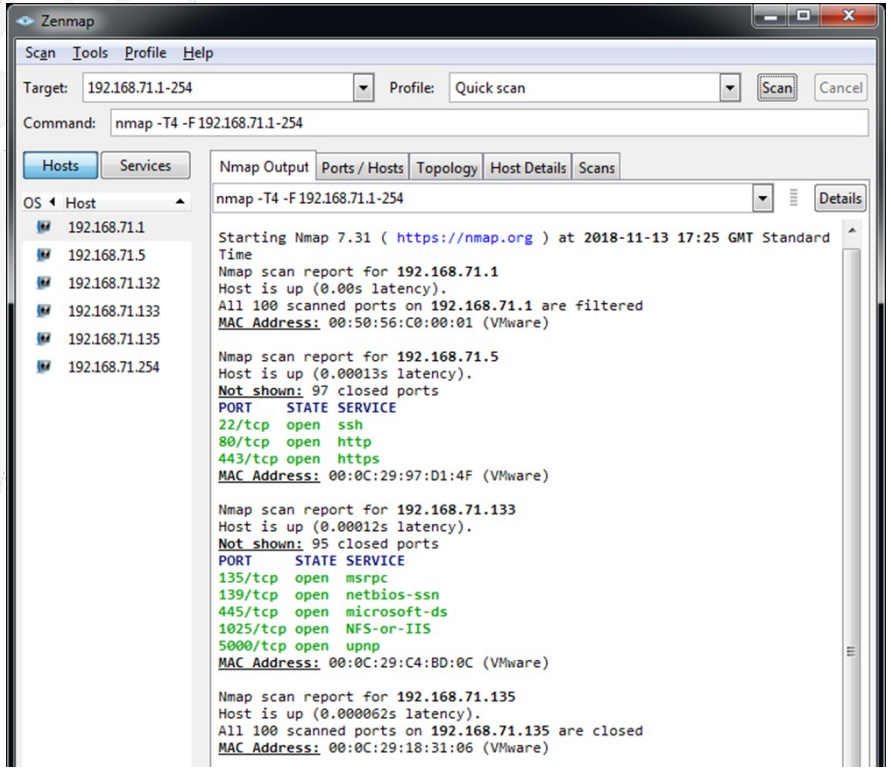



```
ub@ub:~$ iperf -c 10.240.218.51 -u
------------------------------------------------------------
Client connecting to 10.240.218.51, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.240.216.172 port 59409 connected with 10.240.218.51 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0161 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0163 sec  1.25 MBytes  1.05 Mbits/sec   0.045 ms 0/895 (0%)
ub@ub:~$
```


ub@ub: ~
ub@ub:~$ iperf -s -u

Server listening on UDP port 5001
UDP buffer size: 208 KByte (default)
----
[1] local 10.240.218.51 port 5001 connected with 10.240.216.172 port 59409
[ID] Interval   Transfer   Bandwidth   Jitter   Lost/Total Datagrams
[1] 0.0000-10.0163 sec 1.25 MBytes 1.05 Mbits/sec   0.046 ms 0/895 (0%)

Network Scanning: Nmap



```
┌──(ka㉿kali)-[~]
└─$ nmap -sT -p 80 10.240.217.64
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-17 11:27 GMT
Nmap scan report for nas-10-240-217-64.york.ac.uk (10.240.217.64)
Host is up (0.00055s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

```
┌──(ka㉿kali)-[~]
└─$ nmap -sT -p 80 10.240.218.51
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-17 11:27 GMT
Nmap scan report for nas-10-240-218-51.york.ac.uk (10.240.218.51)
Host is up (0.00055s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

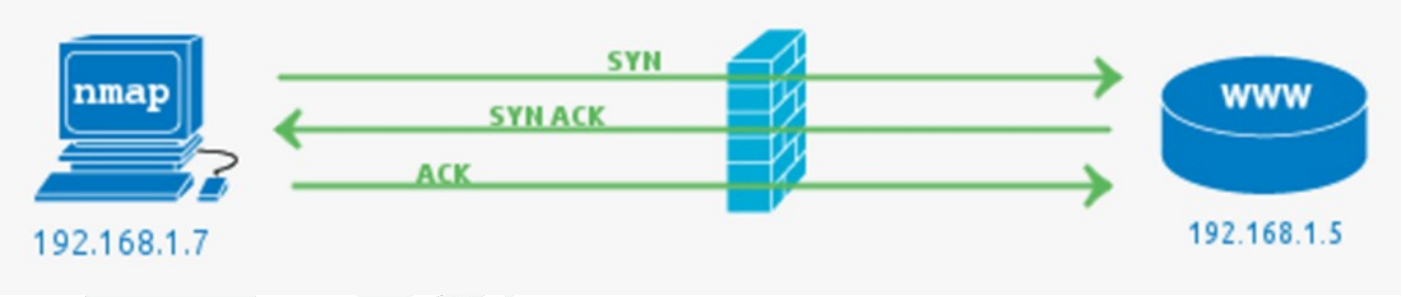
Network Scanning using NetCat

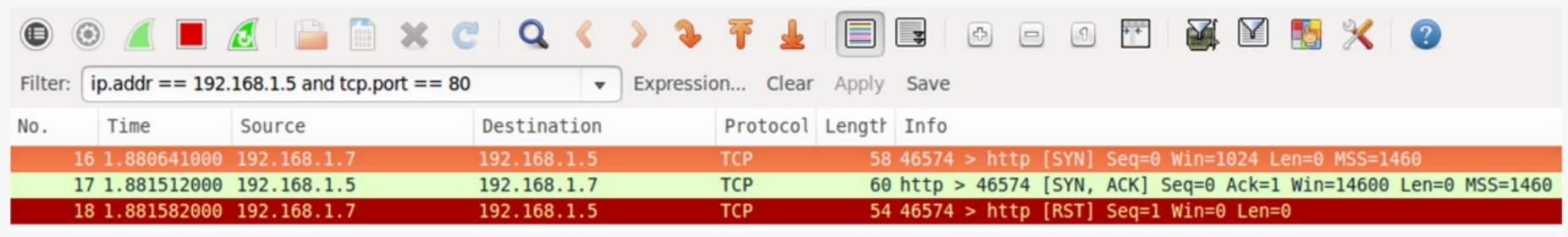
```
┌──(ka㉿kali)-[~]
└─$ nc -v -w2 -z 10.240.218.131 1-255
nas-10-240-218-131.york.ac.uk [10.240.218.131] 80 (http) open
nas-10-240-218-131.york.ac.uk [10.240.218.131] 22 (ssh) open

┌──(ka㉿kali)-[~]
└─$
```

```
┌──(ka㉿kali)-[~]
└─$ nc -v -n 10.240.218.131 22
(UNKNOWN) [10.240.218.131] 22 (ssh) open
SSH-2.0-OpenSSH_9.4p1 Debian-1

┌──(ka㉿kali)-[~]
└─$
```

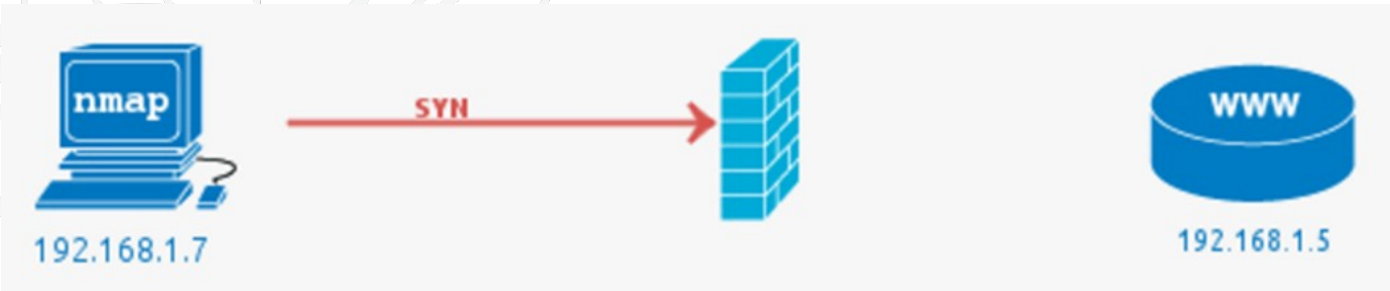
```
┌──(ka㉿kali)-[~]
└─$ nc 10.240.218.131 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx/1.24.0
Date: Wed, 14 Feb 2024 15:34:12 GMT
Content-Type: text/html
Content-Length: 10705
Last-Modified: Sat, 09 Sep 2023 19:03:01 GMT
Connection: close
ETag: "64fcc165-29d1"
Accept-Ranges: bytes


┌──(ka㉿kali)-[~]
└─$
```

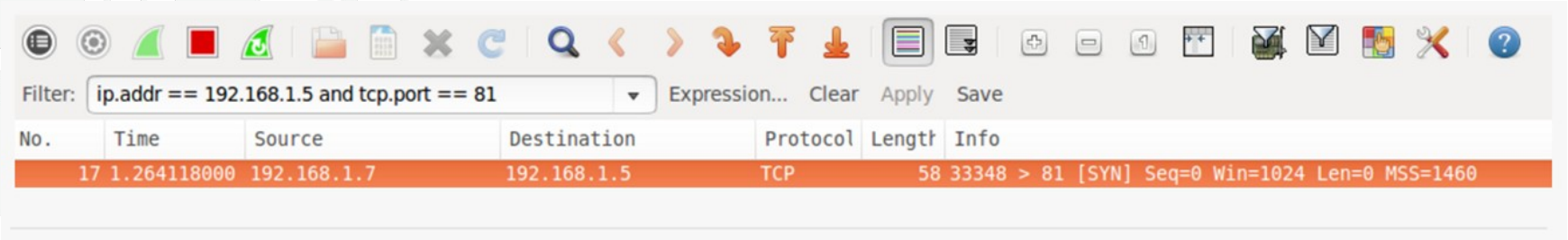
Network Scanning: Open ports



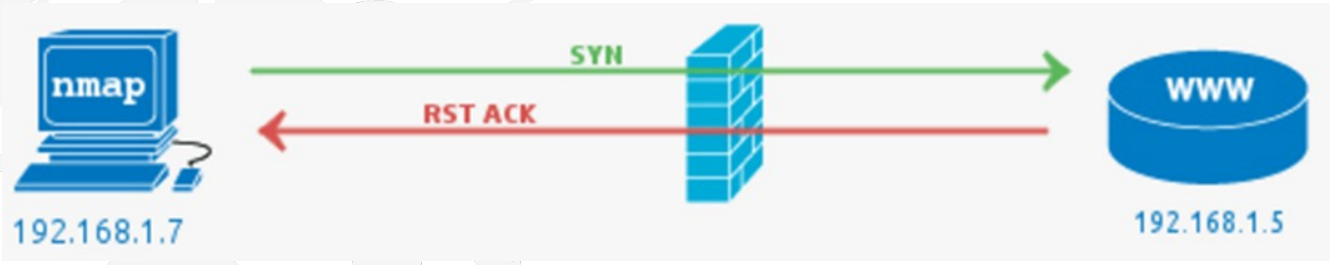


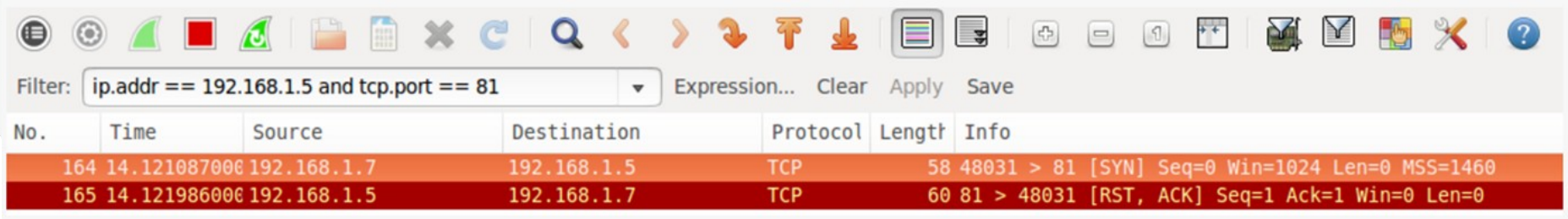
Network Scanning: Filtered ports

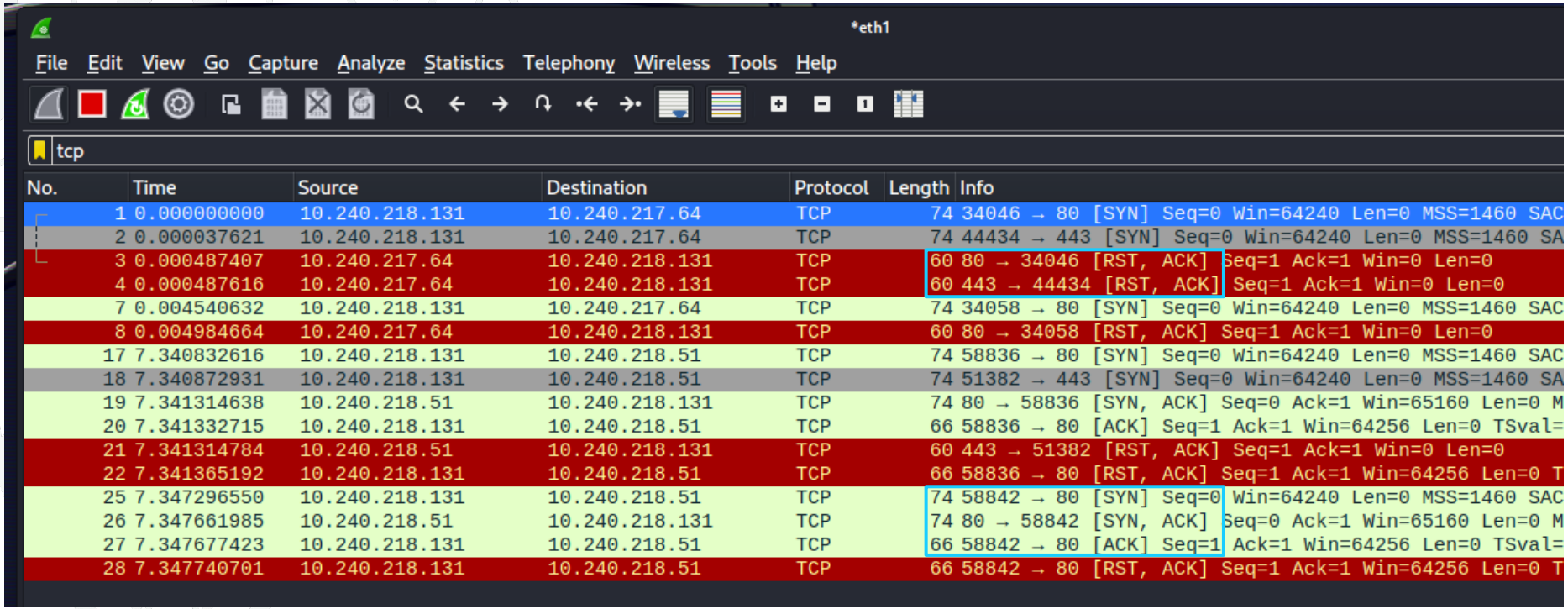




Network Scanning: Closed ports



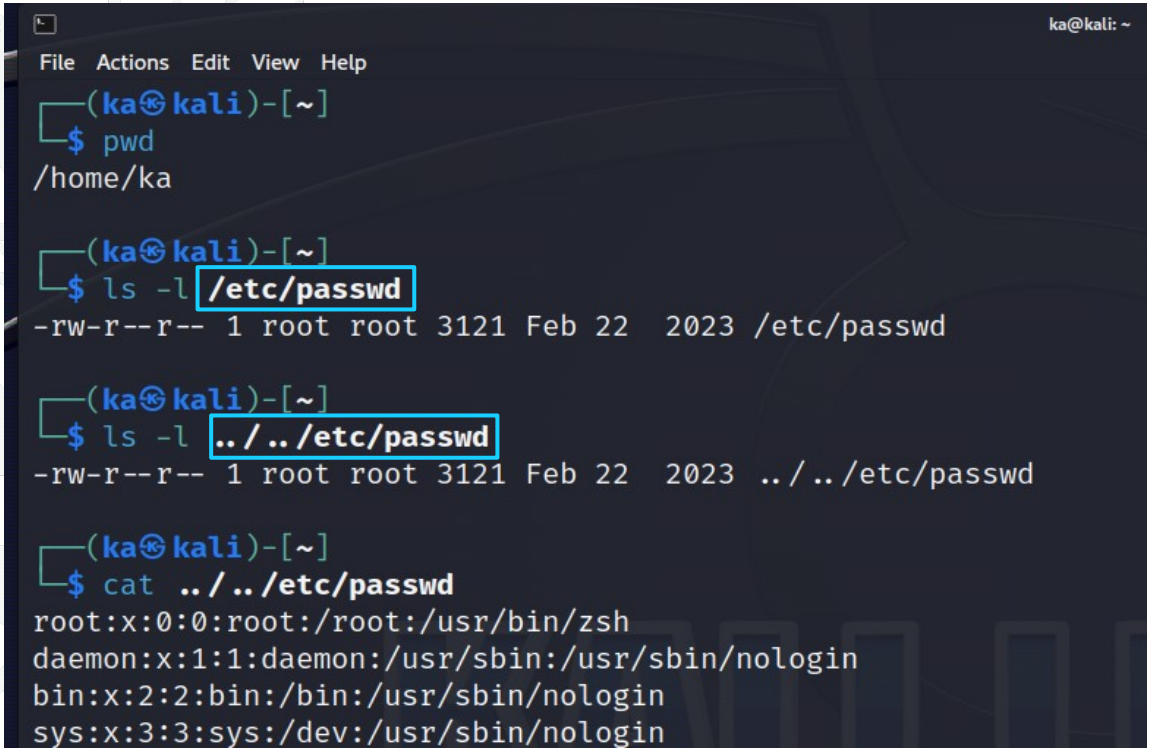



*eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
No. Time Source Destination Protocol Length Info
1 0.000000000 10.240.218.131 10.240.217.64 TCP 74 34046 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SAC
2 0.000037621 10.240.218.131 10.240.217.64 TCP 74 44434 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SA
3 0.000487407 10.240.217.64 10.240.218.131 TCP 60 80 → 34046 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4 0.000487616 10.240.217.64 10.240.218.131 TCP 60 443 → 44434 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7 0.004540632 10.240.218.131 10.240.217.64 TCP 74 34058 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SAC
8 0.004984664 10.240.217.64 10.240.218.131 TCP 60 80 → 34058 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17 7.340832616 10.240.218.131 10.240.218.51 TCP 74 58836 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SAC
18 7.340872931 10.240.218.131 10.240.218.51 TCP 74 51382 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SA
19 7.341314638 10.240.218.51 10.240.218.131 TCP 74 80 → 58836 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 M
20 7.341332715 10.240.218.131 10.240.218.51 TCP 66 58836 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=
21 7.341314784 10.240.218.51 10.240.218.131 TCP 60 443 → 51382 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22 7.341365192 10.240.218.131 10.240.218.51 TCP 66 58836 → 80 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 T
25 7.347296550 10.240.218.131 10.240.218.51 TCP 74 58842 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SAC
26 7.347661985 10.240.218.51 10.240.218.131 TCP 74 80 → 58842 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 M
27 7.347677423 10.240.218.131 10.240.218.51 TCP 66 58842 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=
28 7.347740701 10.240.218.131 10.240.218.51 TCP 66 58842 → 80 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 T

Exploiting Directory Traversal Vulnerabilities

absolute path

relative path



Exploiting Directory Traversal Vulnerabilities

Pretty Raw Hex

```
1  GET /image?filename=37.jpg HTTP/2
2  Host: 0a79009903c97729824e065e00db00bf.web-security-academy.net
3  Cookie: session=gN7XaPBOYZkftbD09TFQRoUOMmp2JpYf
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
5  Accept: image/avif,image/webp,*/*
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer:
   https://0a79009903c97729824e065e00db00bf.web-security-academy.net/product?productId=1
9  Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Te: trailers
```

Attackers can modify the path of requested resources

Supplementary Reading

- OWASP - Path Traversal

https://owasp.org/www-community/attacks/Path_Traversal

- Nmap Tutorial

https://hackertarget.com/nmap-tutorial/

- Understanding Nmap Scan with Wireshark

https://www.hackingarticles.in/understanding-nmap-scan-wireshark/

UNIVERSITY of York

Steganography: Hiding Data in Images

Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

Outline

- What is Steganography?
- Stegosystems
  - General models and features
- Hiding an image in another image
  - Extracting the hidden image
  - High-level algorithm and functions

What is Steganography?

- Steganography is the **hiding** of data in an image or other digital artefact

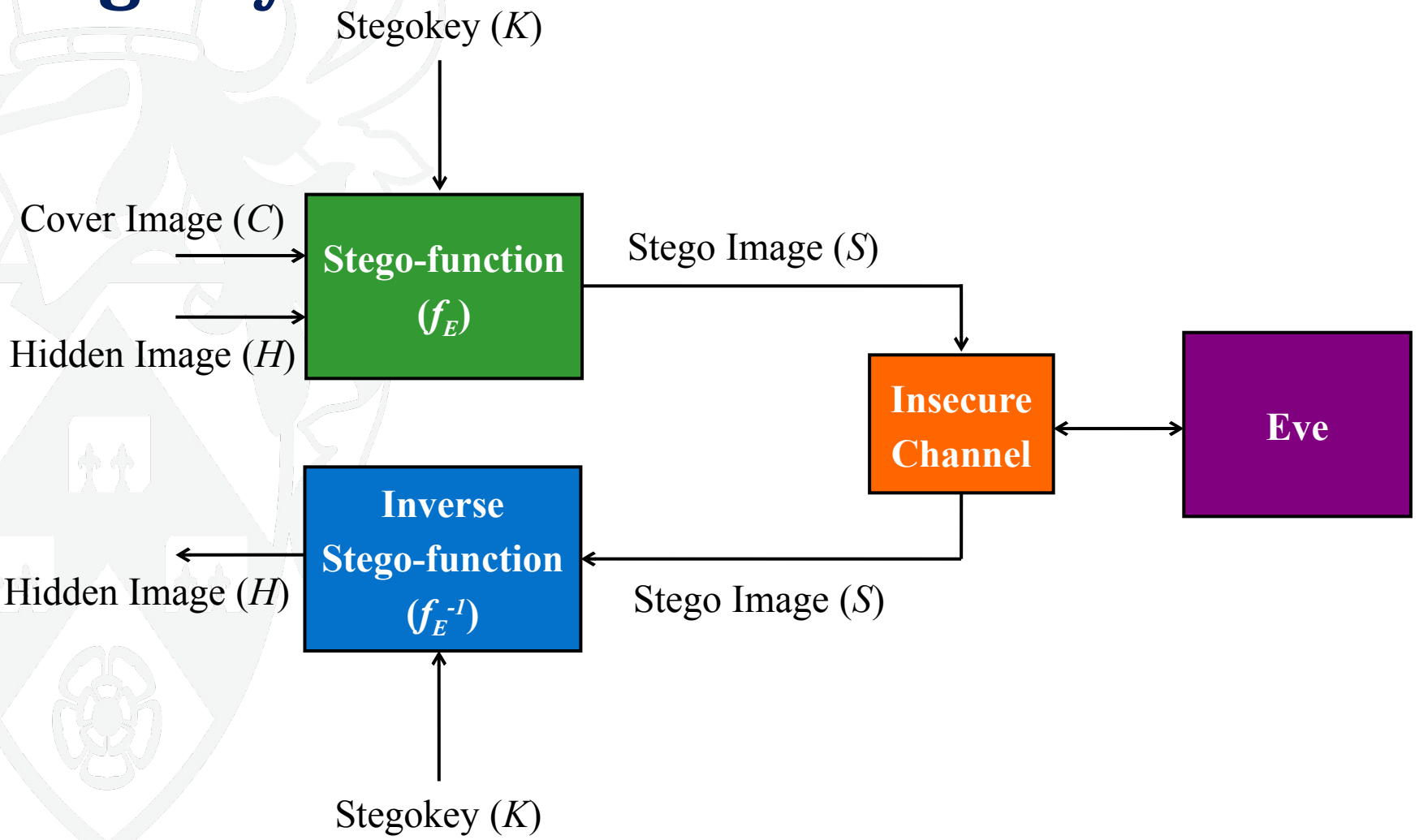
hidden image

cover image

stego image

- Hiding data requires some **math** and **coding**

Stego-systems: General Model



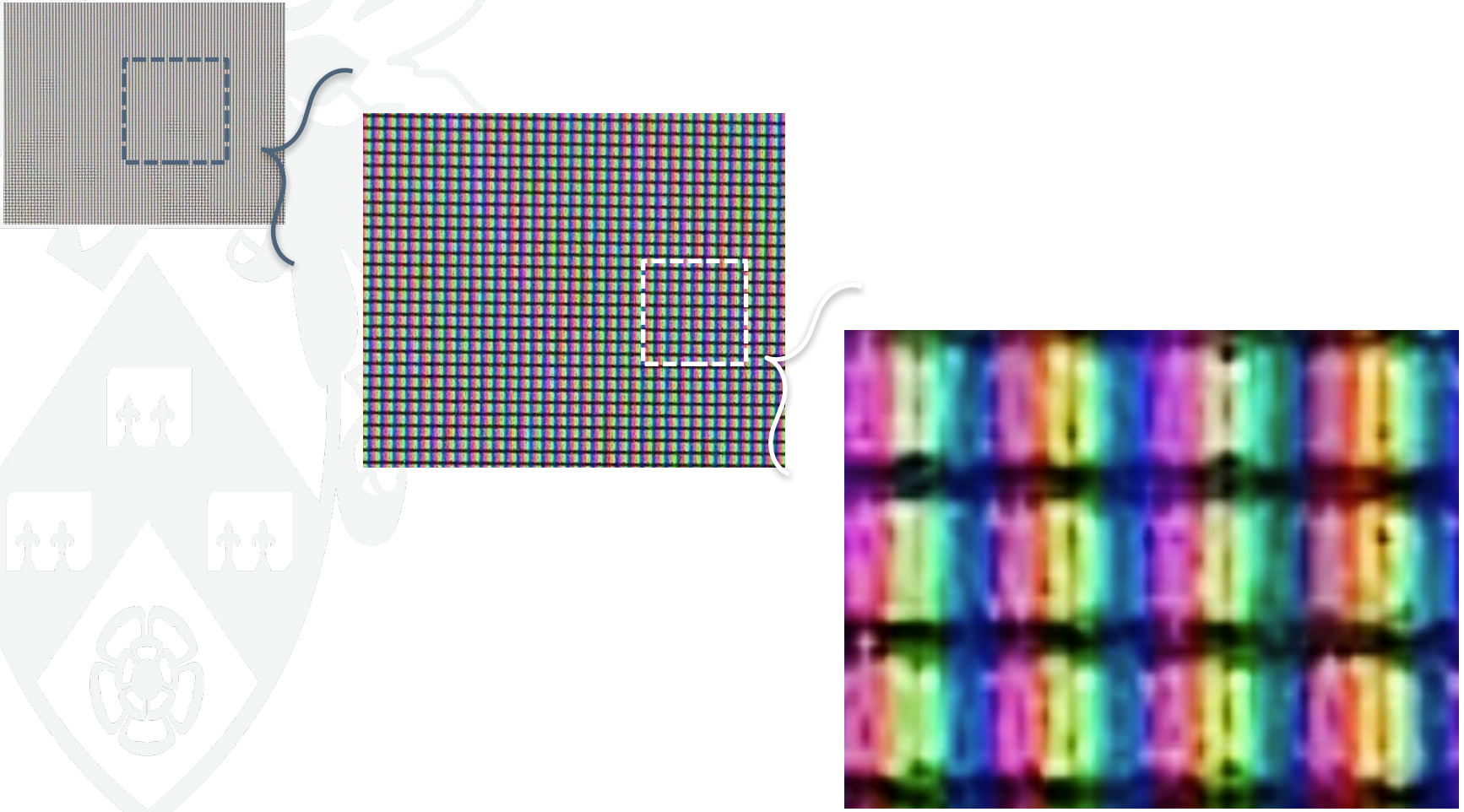
Stego-systems: Considerations

- **Capacity:** How much information can be embedded.

- **Security:** How difficult is for an opponent to tell whether an object has or has not some hidden data (e.g, image or text) embedded.

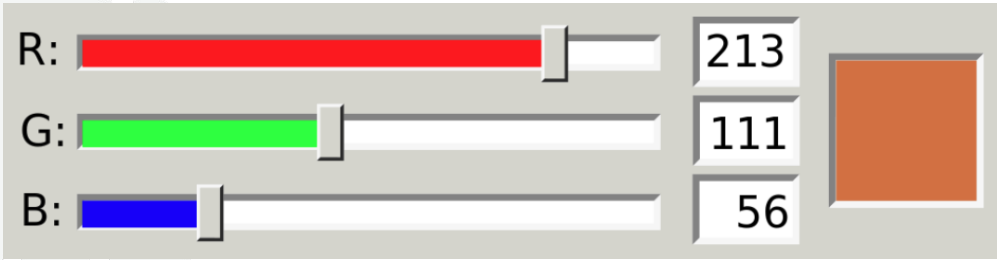
- **Robustness:** How many modifications the stego image (stego text, etc) can endure before the hidden image (hidden text, etc.) is damaged.

Images

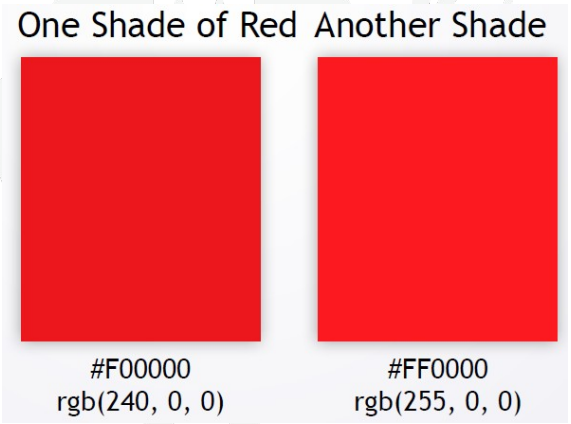


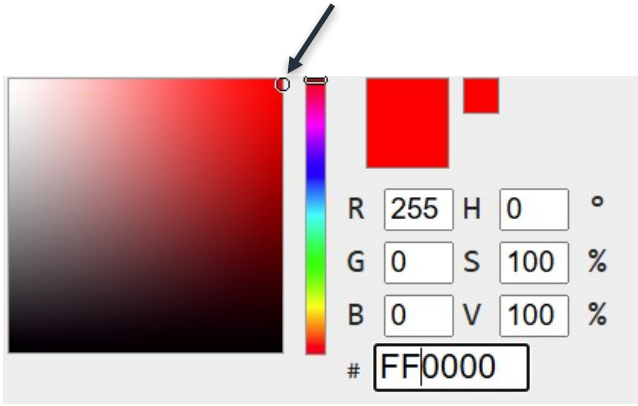
How to Hide Data in Pixels?

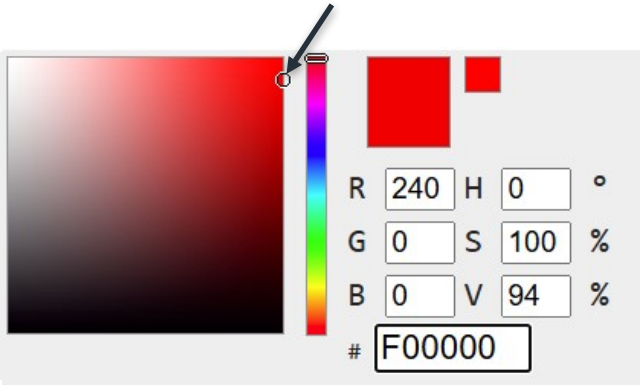
- Pixels have Red, Green, Blue components
  - Each is



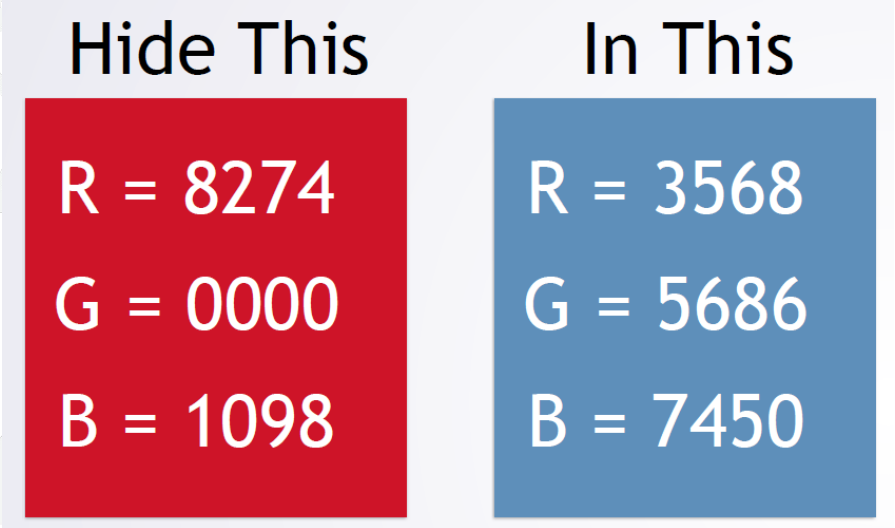
  - Is there a big difference between 240 and 255?







Hide an Image in another Image

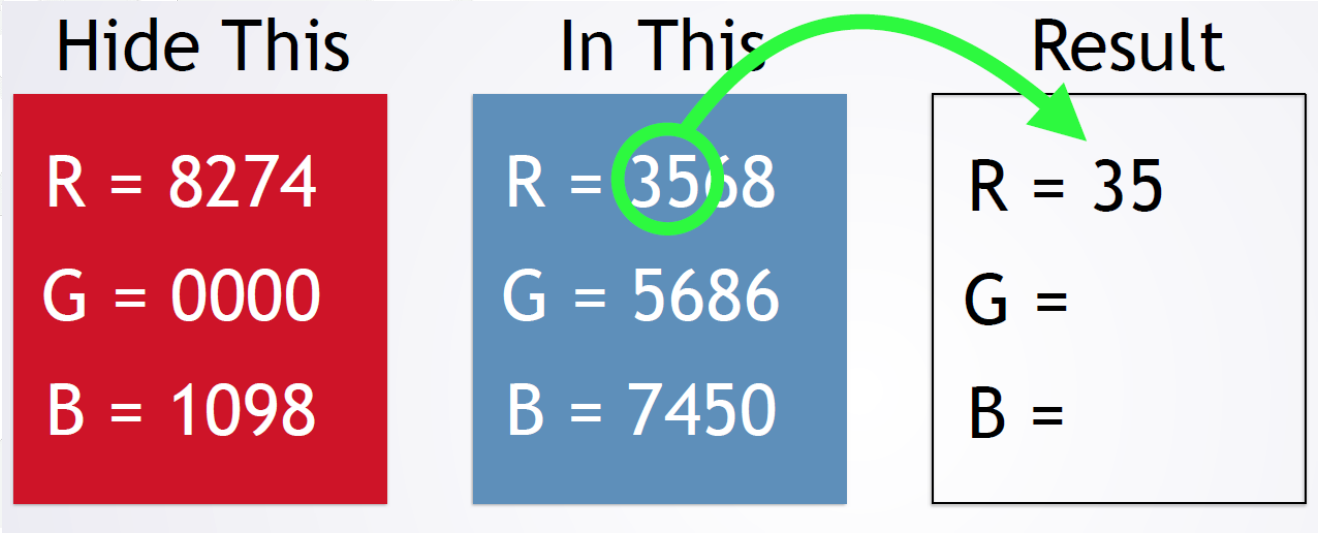


Real RGB values are from 0 to 255 (i.e., 8 bits)

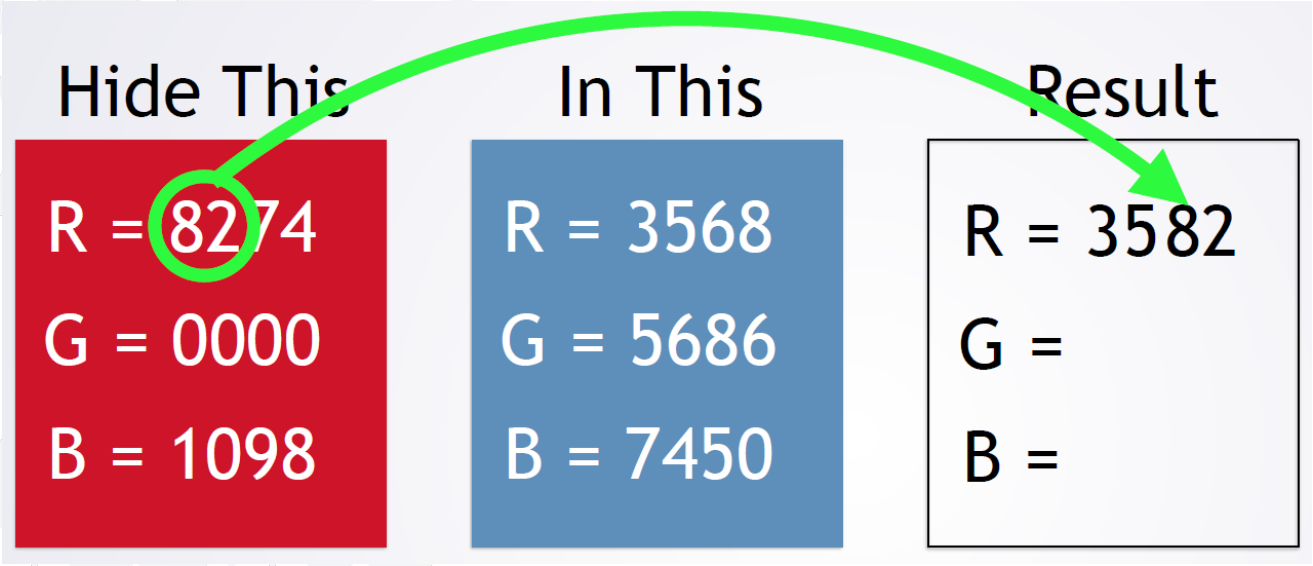
Values here are examples for presentation

To show 2 “high digits” and 2 “low digits”

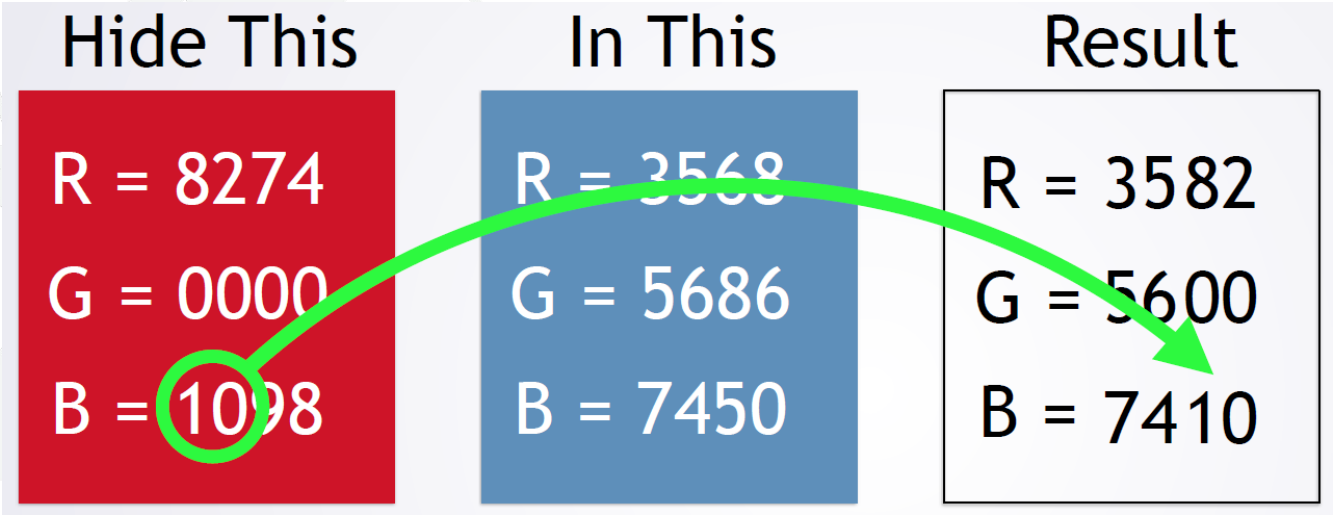
Hide an Image in another Image



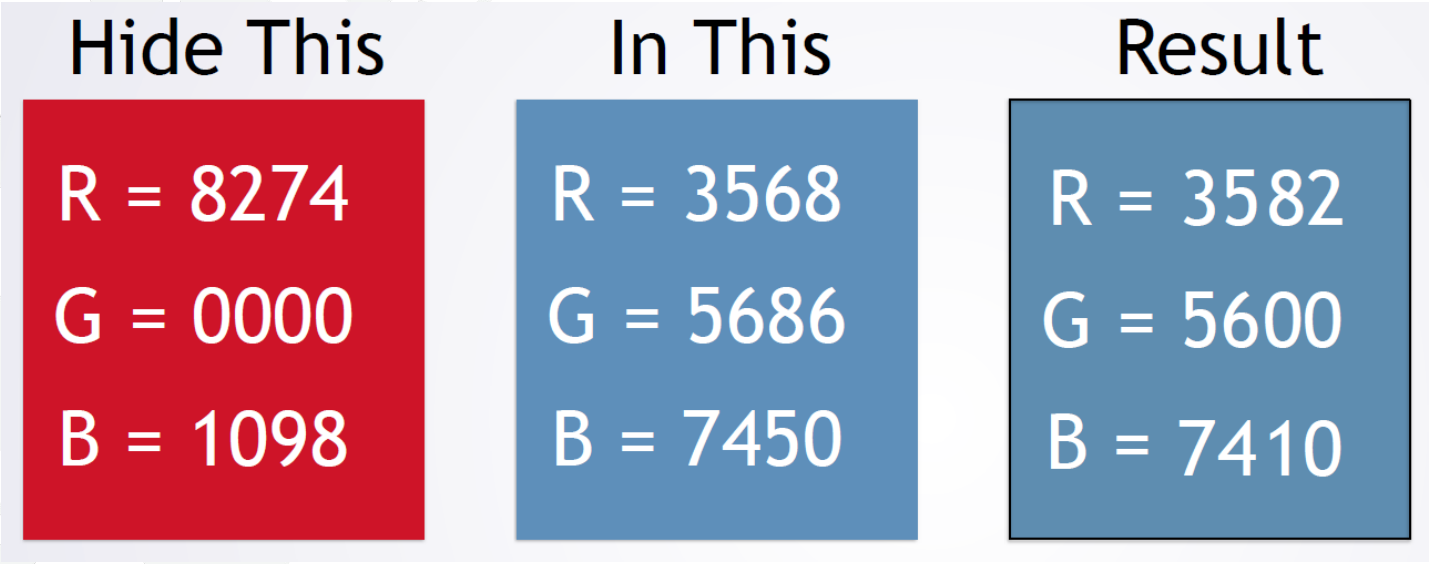
Hide an Image in another Image



Hide an Image in another Image



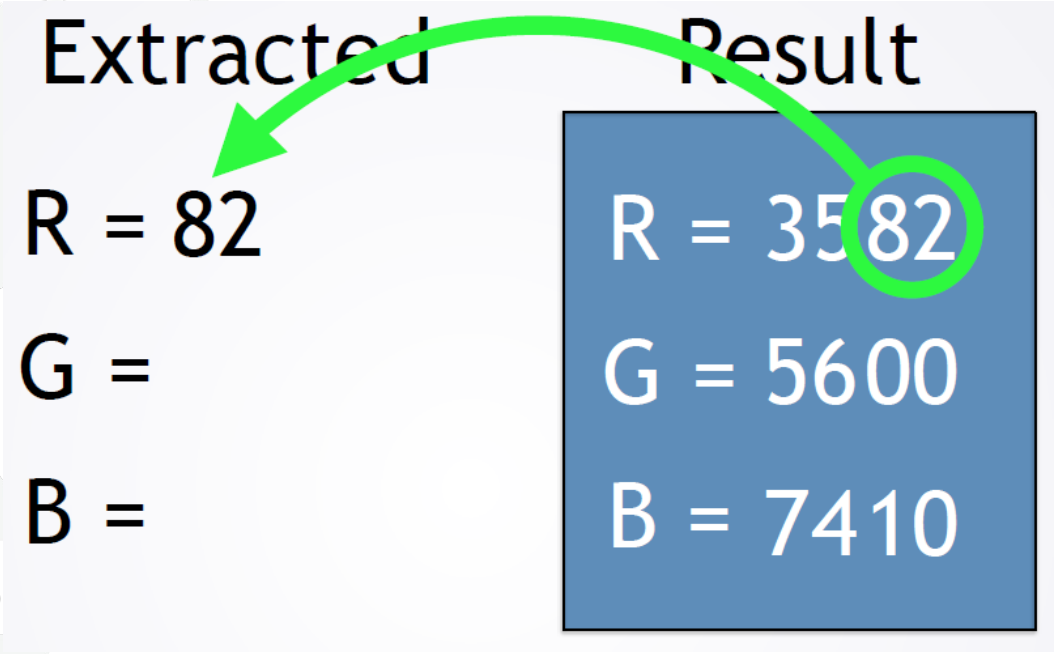
Hide an Image in another Image



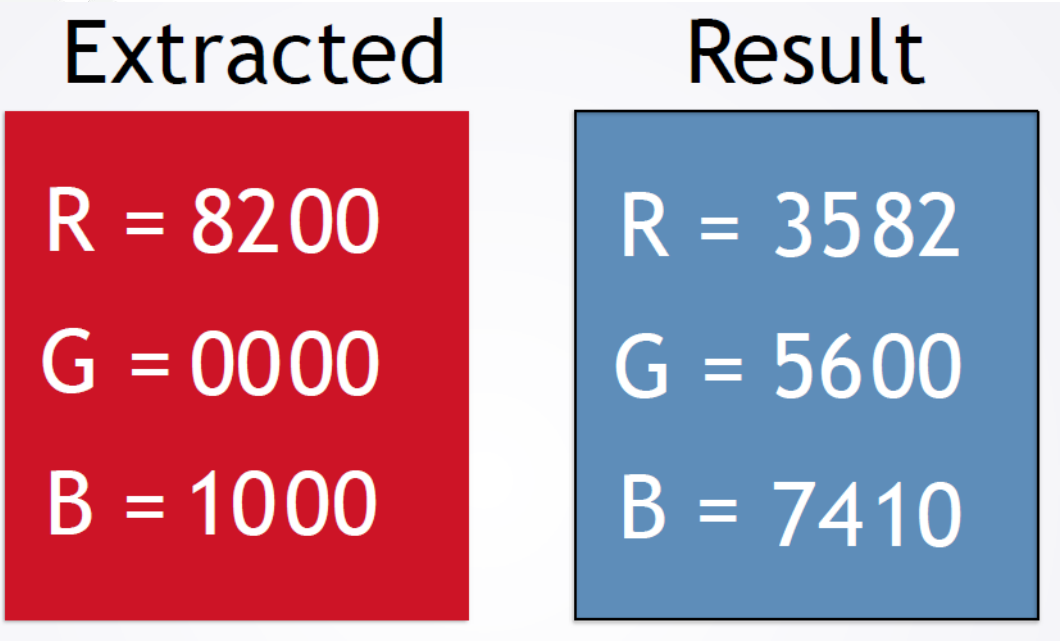
Extract the Hidden Image

Extracted	Result
R =	R = 3582
G =	G = 5600
B =	B = 7410

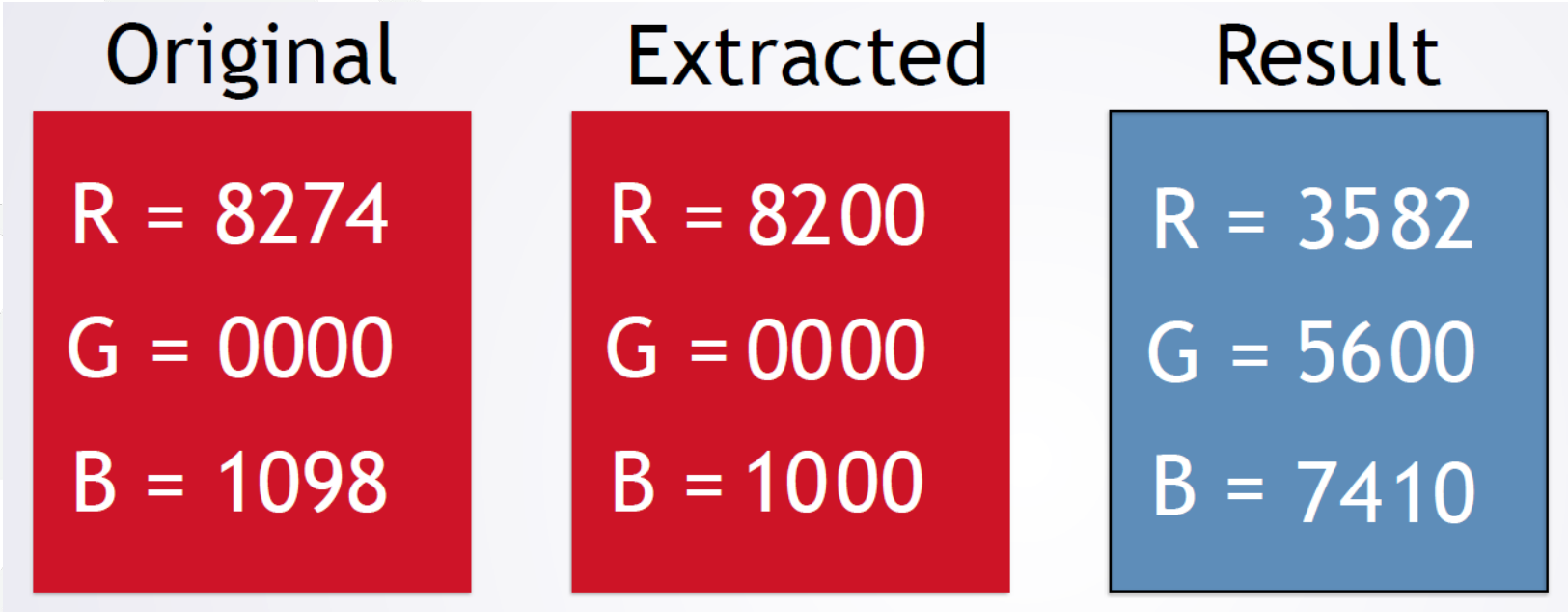
Extract the Hidden Image



Extract the Hidden Image



Extract the Hidden Image



Steganography with Numbers

Hide This 8274 In This 3568

3582

How to do this with math?

Math.floor(8274/100) = 82

Math.floor(3568/100) = 35

35 * 100 + 82 = 3582

2 digits -> 10^2 =100

Steganography with Numbers

Extract hidden message	Result
3582	8200

How to do this with math?

3582 % 100 = 82

82 * 100 = 8200

Steganography with Binary

Hide This

In This

B2

75

10110010

01110101

01111011

Math.floor(10110010/16) = 1011

Math.floor(01110101/16) = 0111

0111 * 16 + 1011 = 01111011

4 digits -> 2^4 =16

Steganography with Binary

Extract hidden message

01111011

Result

10110000

How to do this with math?

01111011 % 16 = 1011

1011 * 16 = 10110000

Steganography with Images

Hide This

You are learning about a new, digital world --- a world in which you can create things like web pages and programs and then share them with your friends or everyone.

01110101

10110111

In This

10110010

Need to do this for **each** pixel, for **R**, **G**, **B** components

Steganography with Images

Hide This

You are learning about a new, digital world --- a world in which you can create things like web pages and programs and then share them with your friends or everyone.

For each pixel

```
Red = Math.floor(red/16) * 16
Green = Math.floor (green/16) * 16
Blue = Math.floor(blue/16) * 16
```

01110101 → 01110000

In This

chop2hide →

Steganography with Images

Hide This

You are learning about a new, digital world --- a world in which you can create things like web pages and programs and then share them with your friends or everyone.

shift

For each pixel

Red = `Math.floor(red/16)`

Green = `Math.floor(green/16)`

Blue = `Math.floor(blue/16)`

10110010 → 00001011

In This

chop2hide

Steganography with Images

Hide This

You are learning about a new, digital world --- a world in which you can create things like web pages and programs and then share them with your friends or everyone.

shift

combine

In This

chop2hide

High-Level Algorithm

```
var start = new SimpleImage("usain.jpg");
var hide = new SimpleImage("skyline.jpg");

start = chop2hide(start);
hide = shift(hide);
var stego = combine(start,hide);
print(stego);
```

Chop2Hide

```
function clearbits(pixval){
    var x = Math.floor(pixval/16) * 16;
    return x;
}
function chop2hide(image){
    for(var px of image.values()){
        px.setRed(clearbits(px.getRed()));
        px.setGreen(clearbits(px.getGreen()));
        px.setBlue(clearbits(px.getBlue()));
    }
    return image;
}
```

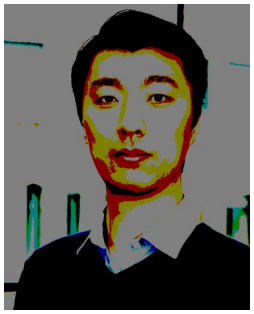
Hidden bits

cover image

rrrr rrrr

hidden image

xxxx xxxx

number of hidden bits	stego image	unhidden image
1 bit rrrr rrrx		
4 bits rrrr xxxx		
7 bits rxxx xxxx		

Cross-Site Scripting (XSS)

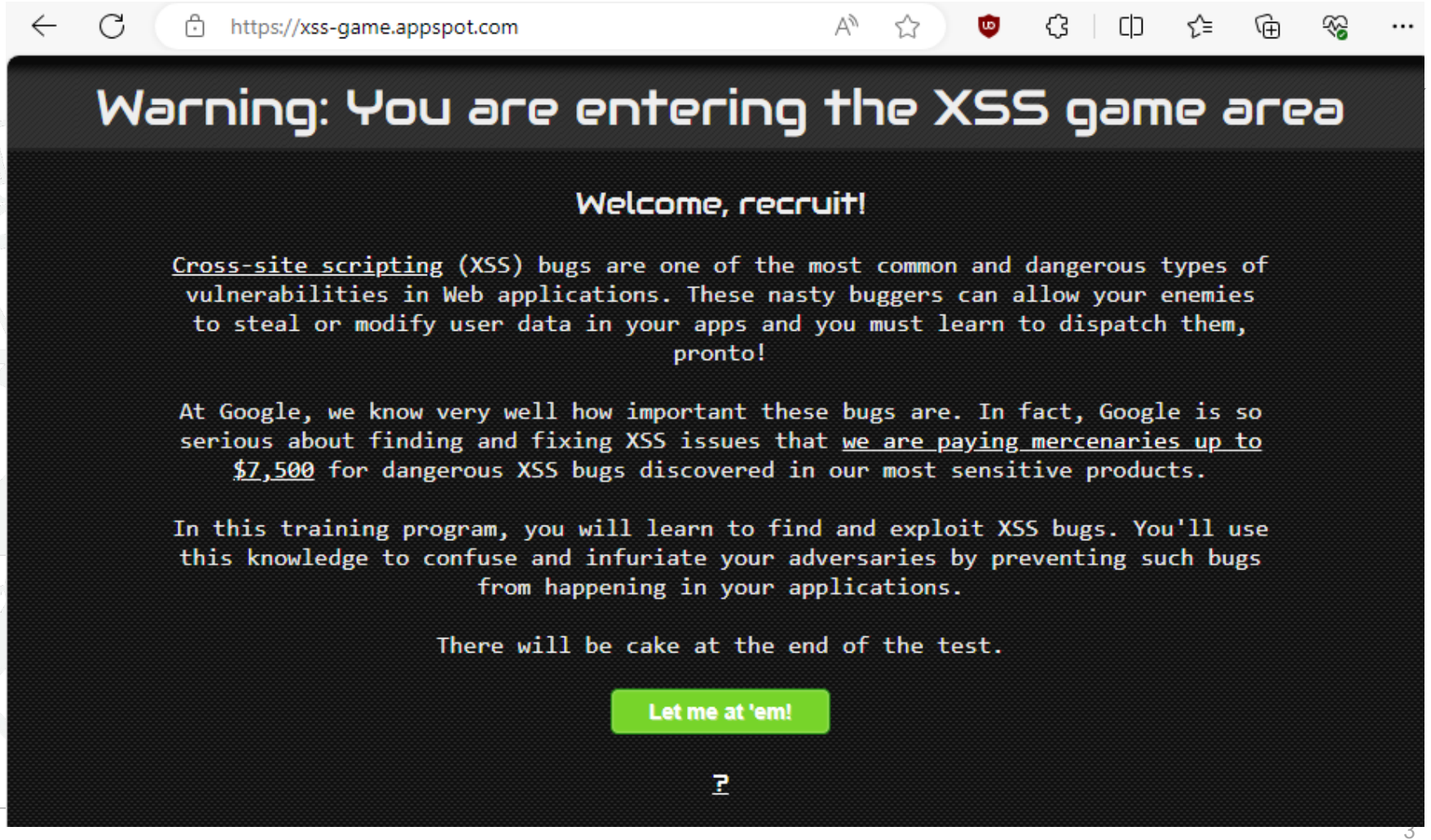
Ethical Hacking, Analysis and Investigation (COM00064H, COM00182M)

Yuchen Zhao, CSE/231, yuchen.zhao@york.ac.uk

Cross-Site Scripting (XSS)

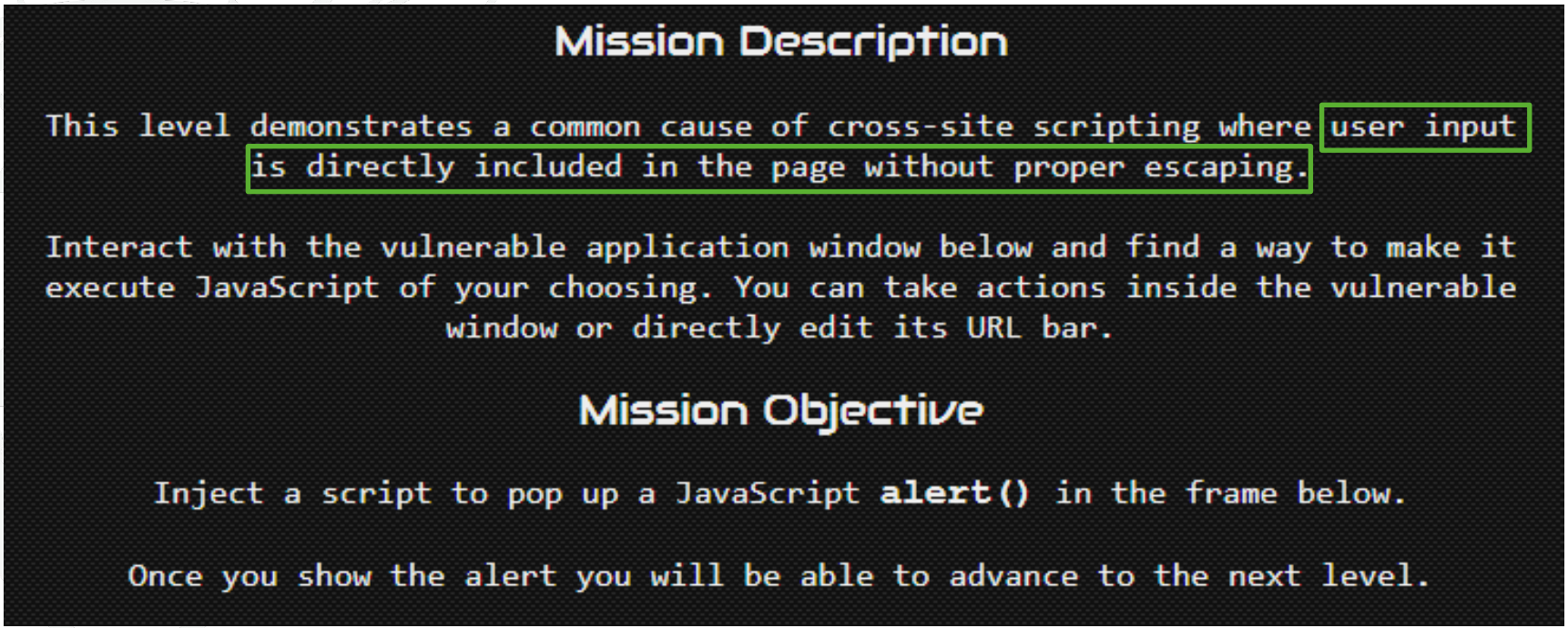
Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites.

- XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a **browser side script**, to a different end user.
- Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.


https://xss-game.appspot.com
Warning: You are entering the XSS game area
Welcome, recruit!
Cross-site scripting (XSS) bugs are one of the most common and dangerous types of vulnerabilities in Web applications. These nasty buggers can allow your enemies to steal or modify user data in your apps and you must learn to dispatch them, pronto!
At Google, we know very well how important these bugs are. In fact, Google is so serious about finding and fixing XSS issues that we are paying mercenaries up to $7,500 for dangerous XSS bugs discovered in our most sensitive products.
In this training program, you will learn to find and exploit XSS bugs. You'll use this knowledge to confuse and infuriate your adversaries by preventing such bugs from happening in your applications.
There will be cake at the end of the test.
Let me at 'em!
?

Level 1

Level 1: Hello, world of XSS




Your Target
I am vulnerable
URL
https://xss-game.appspot.com/level1/frame
Go
FourOrFour
Enter query here...
Search
Target code (toggle)
Hints 0/3 (show)

Exploring the website

Inspecting the Source Code

Target code (toggle)

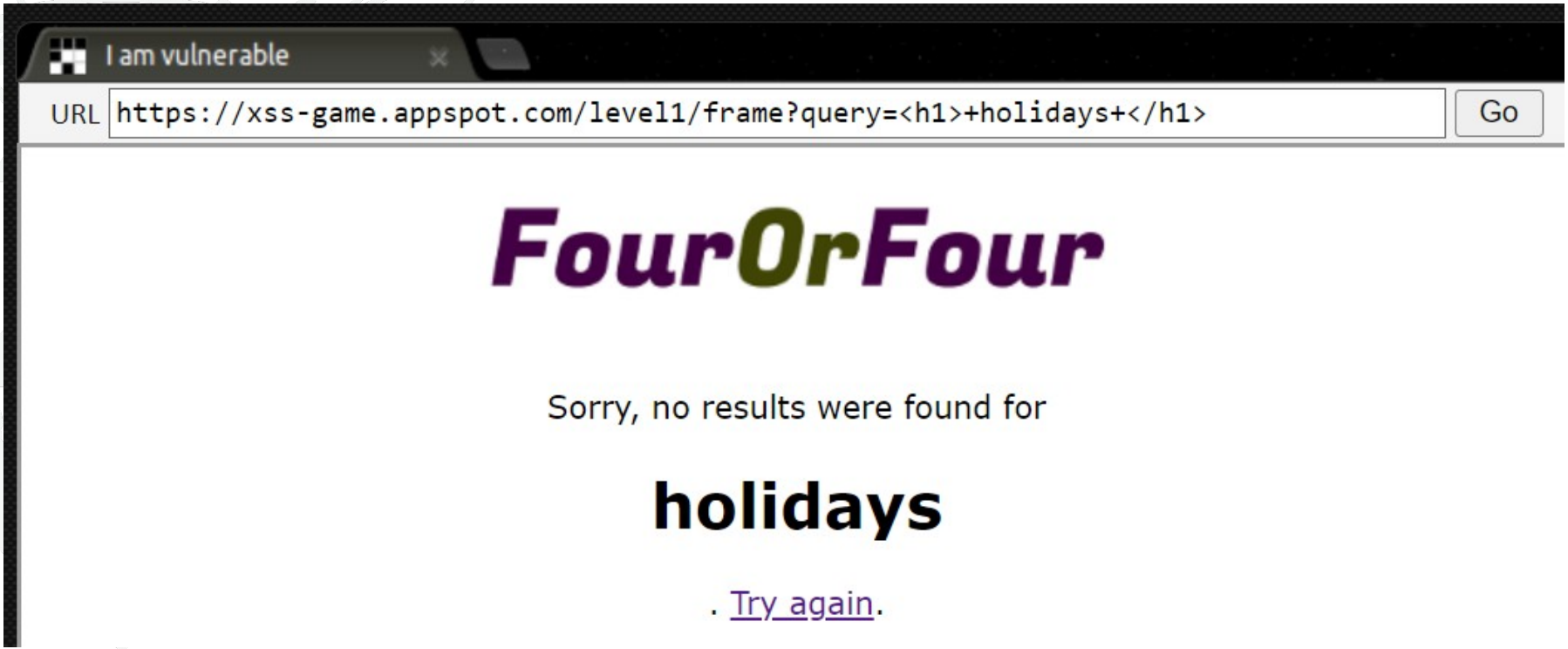
```
16      </div>
17    </body>
18  </html>
19  """
20
21  main_page_markup = """
22  <form action="" method="GET">
23    <input id="query" name="query" value="Enter query here..."
24      onfocus="this.value=''">
25    <input id="button" type="submit" value="Search">
26  </form>
27  """
```

```
44          # Our search engine broke, we found no results :-(
45          message = "Sorry, no results were found for <b>" + query + "</b>."
46          message += " <a href='?'>Try again</a>."
47
48          # Display the results page
49          self.render_string(page_header + message + page_footer)
```

Looking for Vulnerabilities

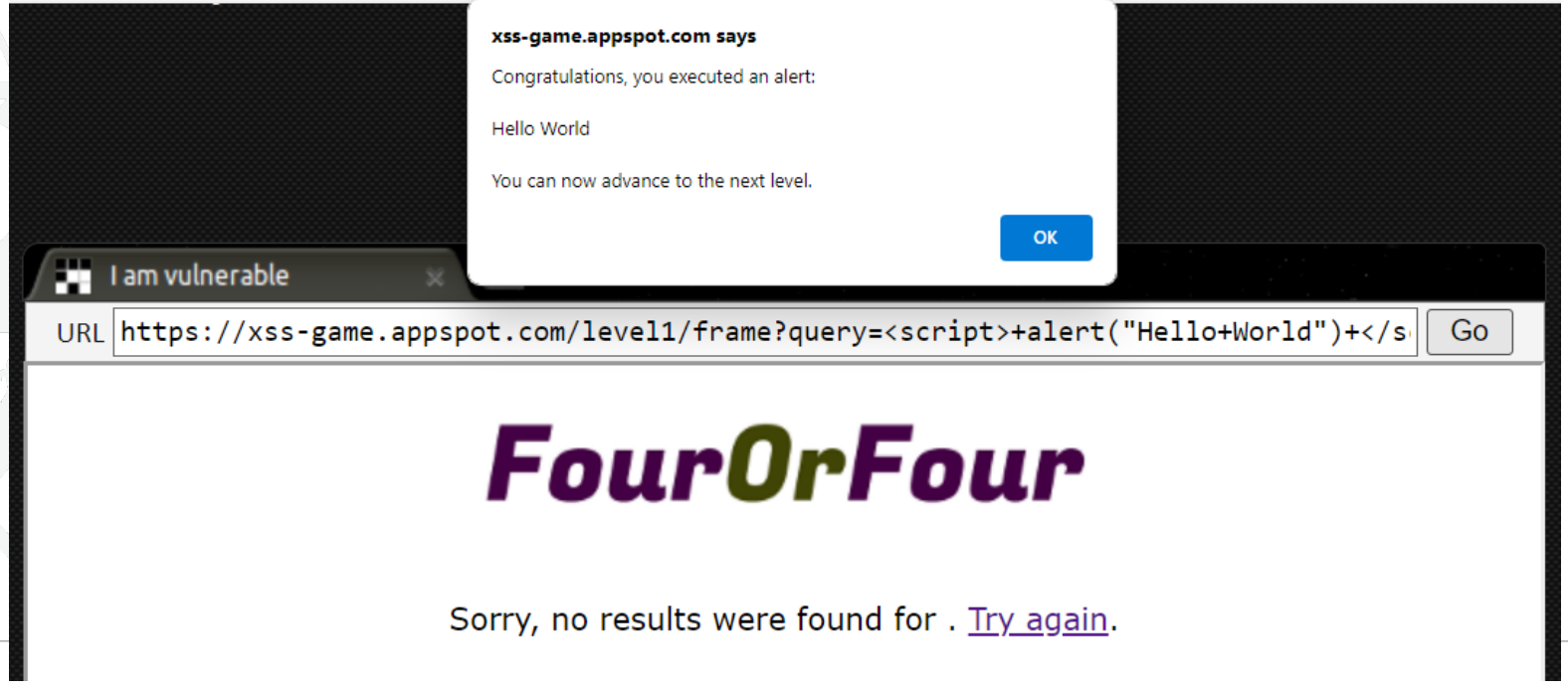
Try to search for: keyword

Try to modify the URL: …frame?query= <h1> keword </h1>



Performing the Attack

Try to search for (or in URL): <script> alert(“keyword”) </script>



Level 2

Level 2: Persistence is key

Mission Description

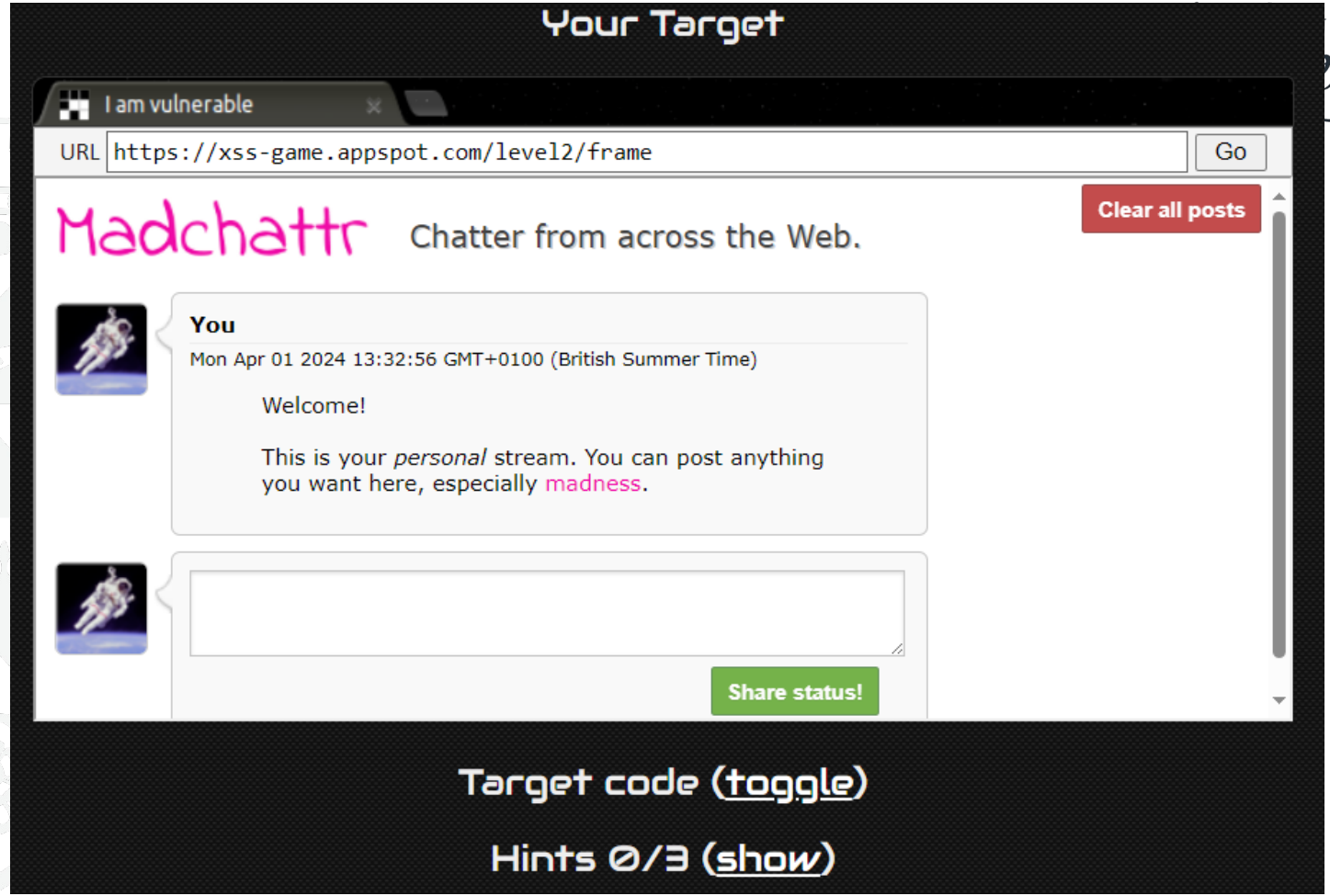
Web applications often keep user data in server-side and, increasingly, client-side databases and later display it to users. No matter where such user-controlled data comes from, it should be handled carefully.

This level shows how easily XSS bugs can be introduced in complex apps.

Mission Objective

Inject a script to pop up an `alert()` in the context of the application.

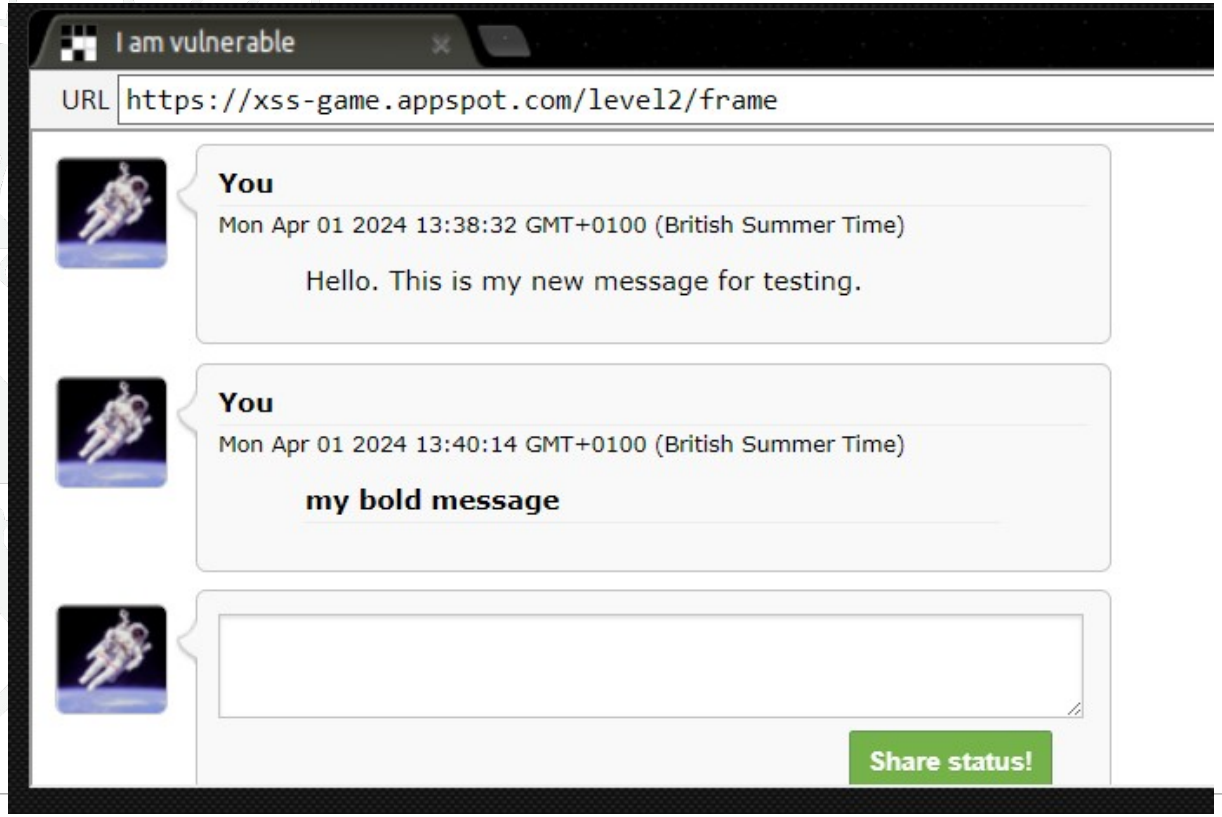
Note: the application saves your posts so if you sneak in code to execute the alert, this level will be solved every time you reload it.


Your Target
I am vulnerable
URL https://xss-game.appspot.com/level2/frame
Go
Madchattr
Chatter from across the Web.
Clear all posts
You
Mon Apr 01 2024 13:32:56 GMT+0100 (British Summer Time)
Welcome!
This is your personal stream. You can post anything you want here, especially madness.
Share status!
Target code (toggle)
Hints 0/3 (show)

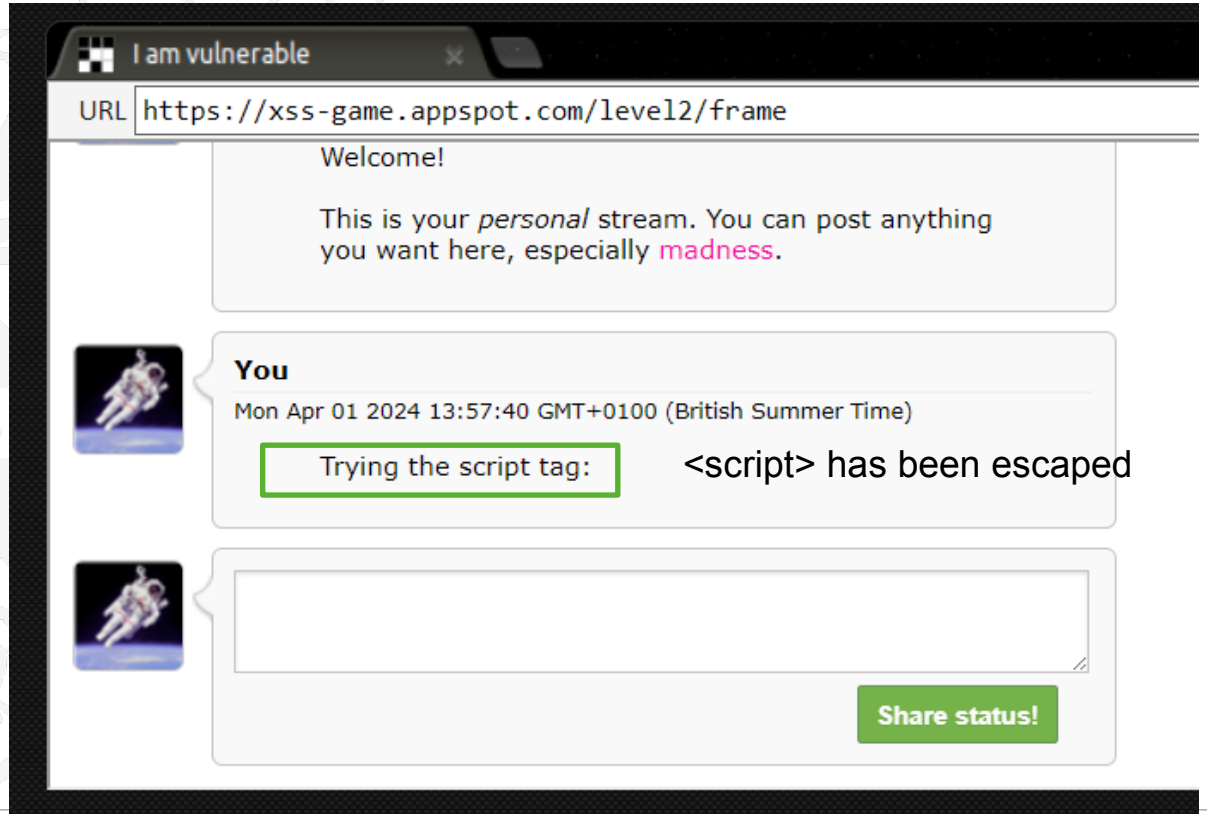
Exploring the Website

Try posting a message, clearing all posts, etc

Try posting a message keyword

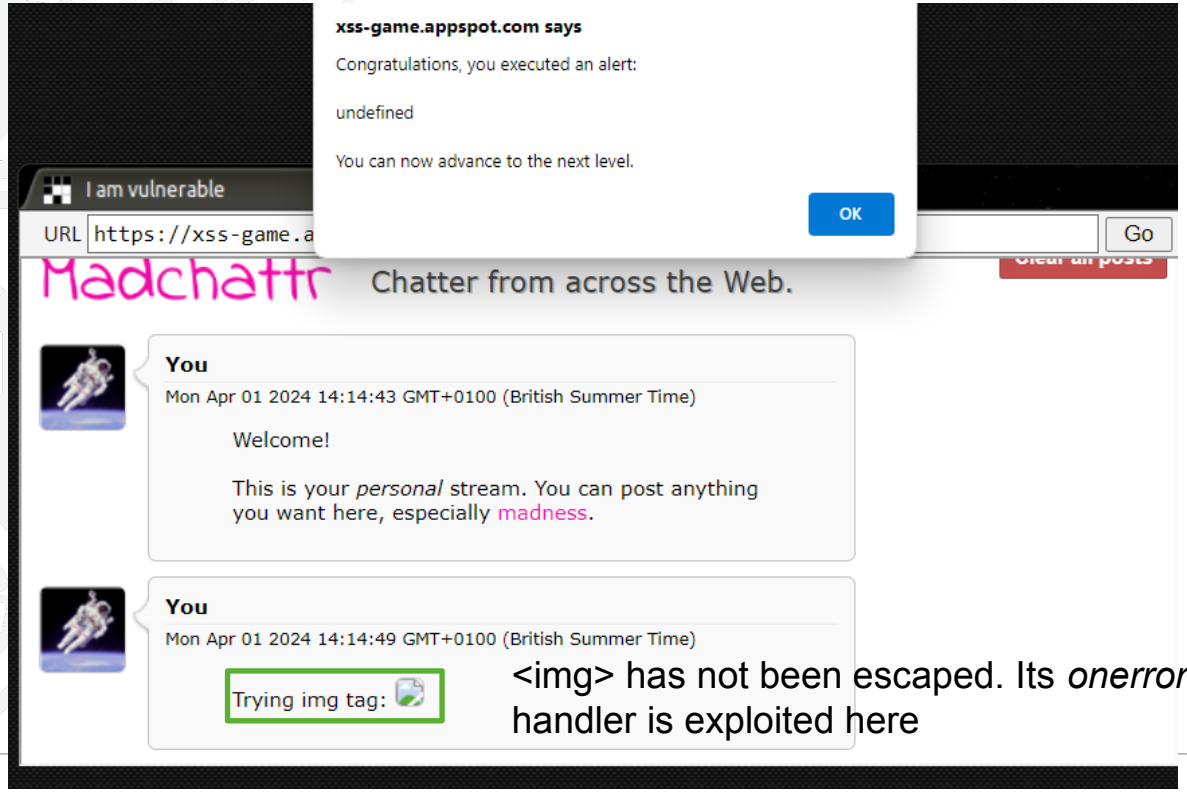


Try posting a message <script> alert('keyword') </script>



Try an element with a JavaScript attribute:

<img src='myimage.jpg' onerror='alert()'>



<img> has not been escaped. Its *onerror* event handler is exploited here

Level 3

Level 3

Mission Description

As you've seen in the previous level, some common JS functions are *execution sinks* which means that they will cause the browser to execute any scripts that appear in their input. Sometimes this fact is hidden by higher-level APIs which use one of these functions under the hood.

The application on this level is using one such hidden sink.

Mission Objective

As before, inject a script to pop up a JavaScript **alert()** in the app.

Since you can't enter your payload anywhere in the application, you will have to manually edit the address in the URL bar below.

Your Target

I am vulnerable

URL https://xss-game.appspot.com/level3/frame#1 Go

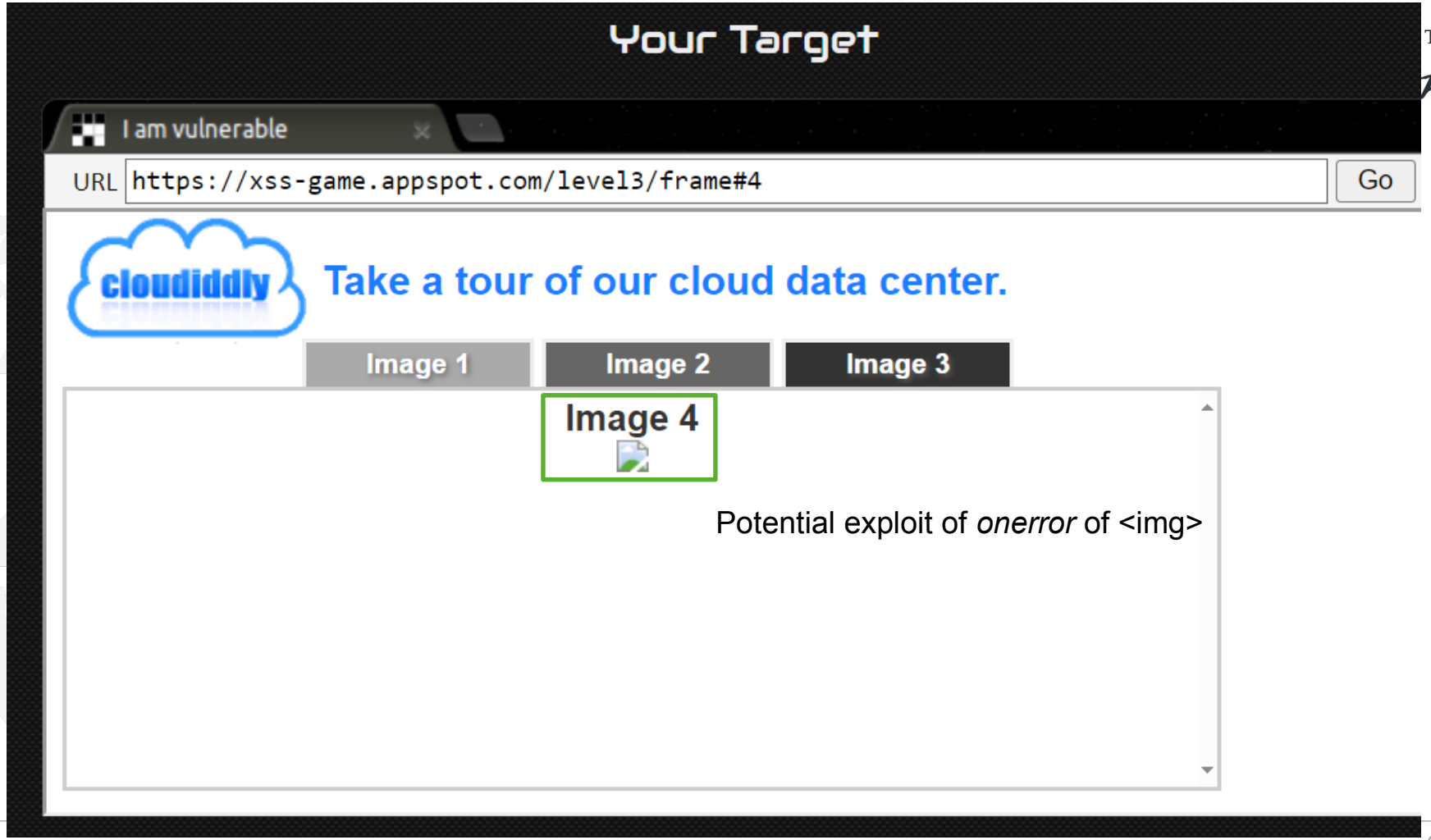
cloudiddly

Take a tour of our cloud data center.

Image 1 | Image 2 | Image 3

Image 1

Your Target

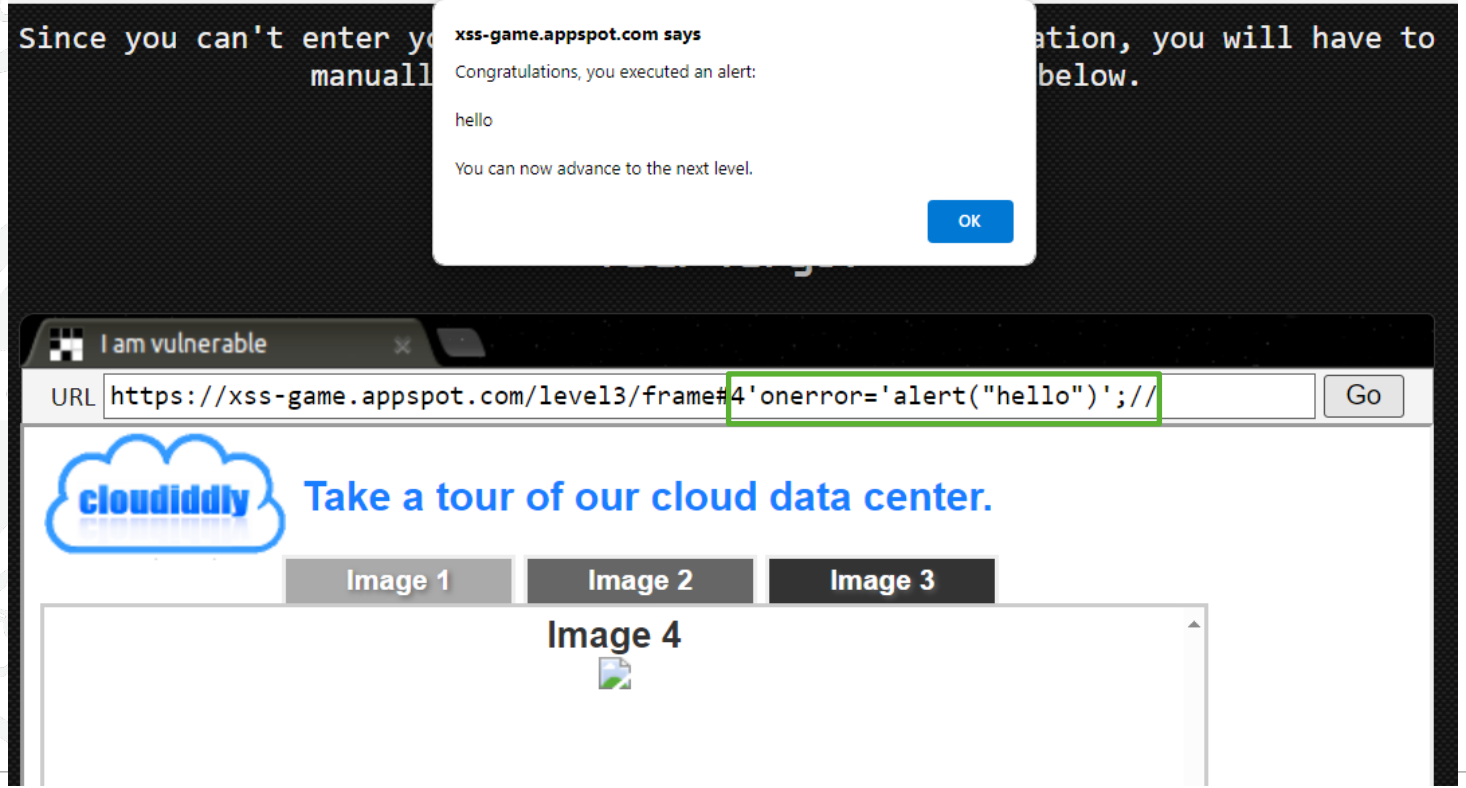


Potential exploit of *onerror* of <img>

Inspecting Source Code

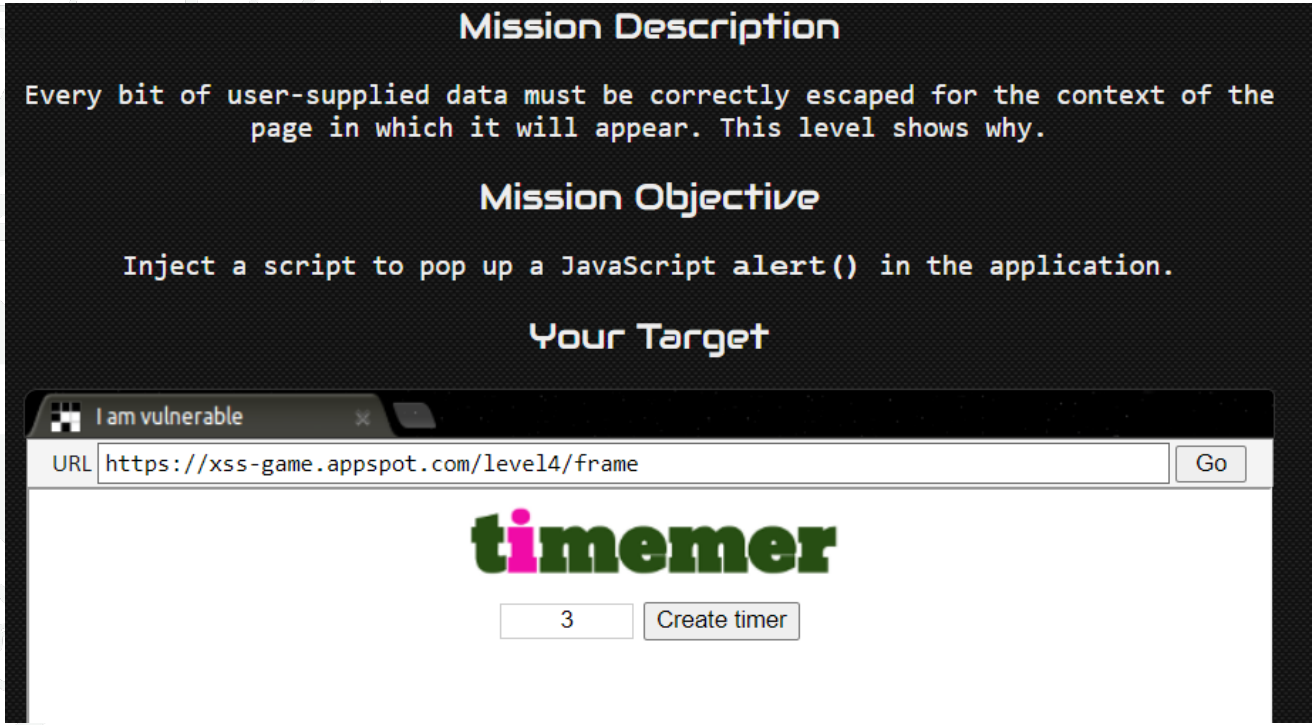
```
13      <script>
14        function chooseTab(num) {
15          // Dynamically load the appropriate image.
16          var html = "Image " + parseInt(num) + "<br>";
17          html += "<img src='/static/level3/cloud" + num + ".jpg' />";
18          $('#tabContent').html(html);
```

Performing the Attack



Level 4

Level 4

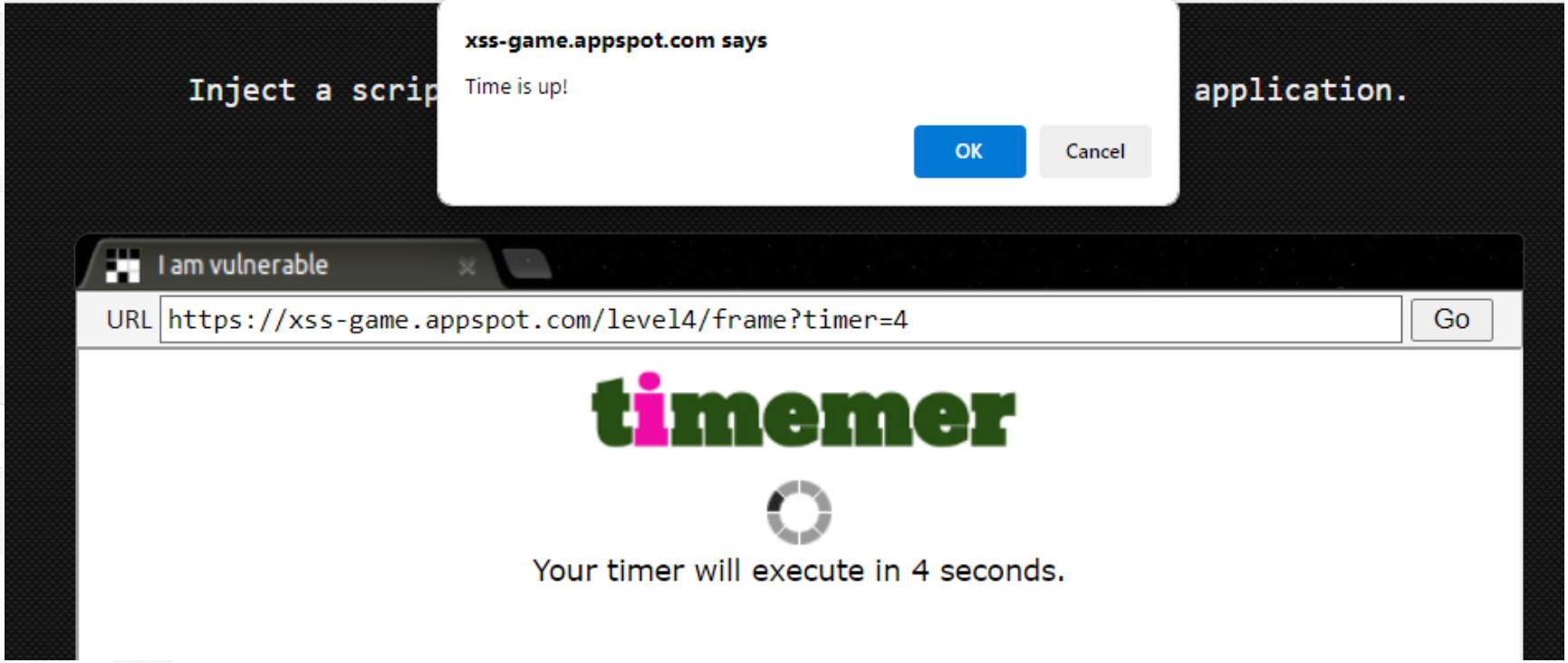

Mission Description
Every bit of user-supplied data must be correctly escaped for the context of the page in which it will appear. This level shows why.
Mission Objective
Inject a script to pop up a JavaScript alert() in the application.
Your Target
I am vulnerable
URL
https://xss-game.appspot.com/level4/frame
Go
timemer
3
Create timer

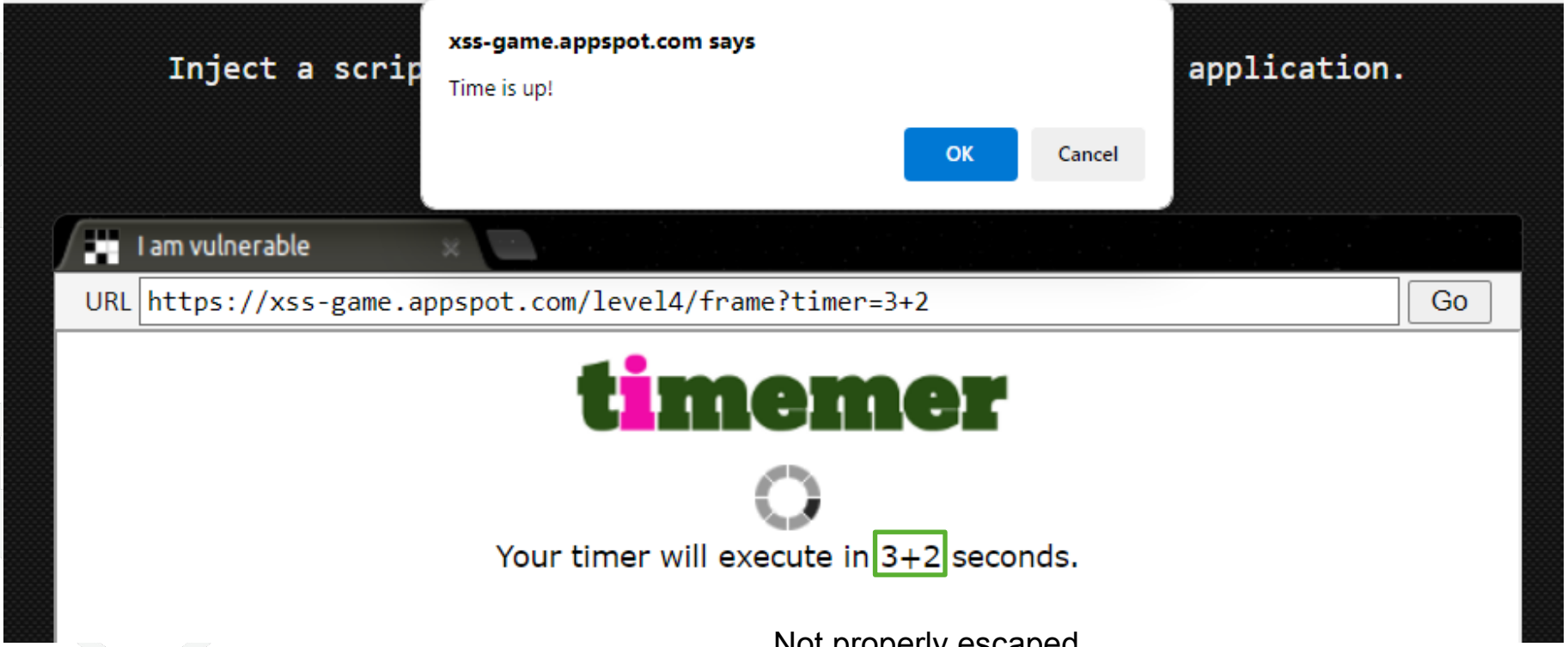
Inspecting the Source Code

index.html | level.py | timer.html

```
1  <!doctype html>
2  <html>
3    <head>
4      <!-- Internal game scripts/styles, mostly boring stuff -->
5      <script src="/static/game-frame.js"></script>
6      <link rel="stylesheet" href="/static/game-frame-styles.css" />
7    </head>
8
9    <body id="level4">
10     <img src="/static/logos/level4.png" />
11     <br>
12     <form action="" method="GET">
13       <input id="timer" name="timer" value="3">
14       <input id="button" type="submit" value="Create timer"> </form>
15     </form>
16   </body>
17 </html>
```

Exploring the Website



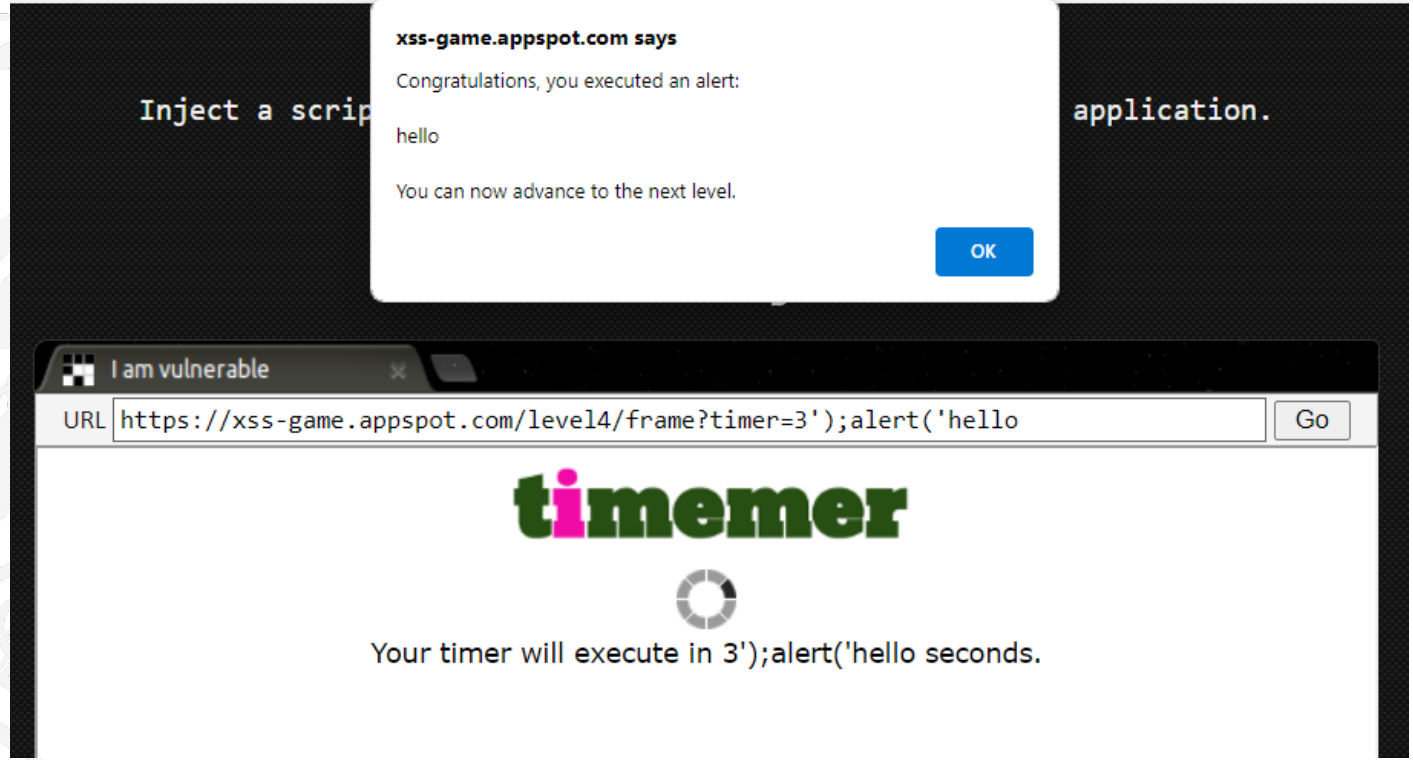


Not properly escaped

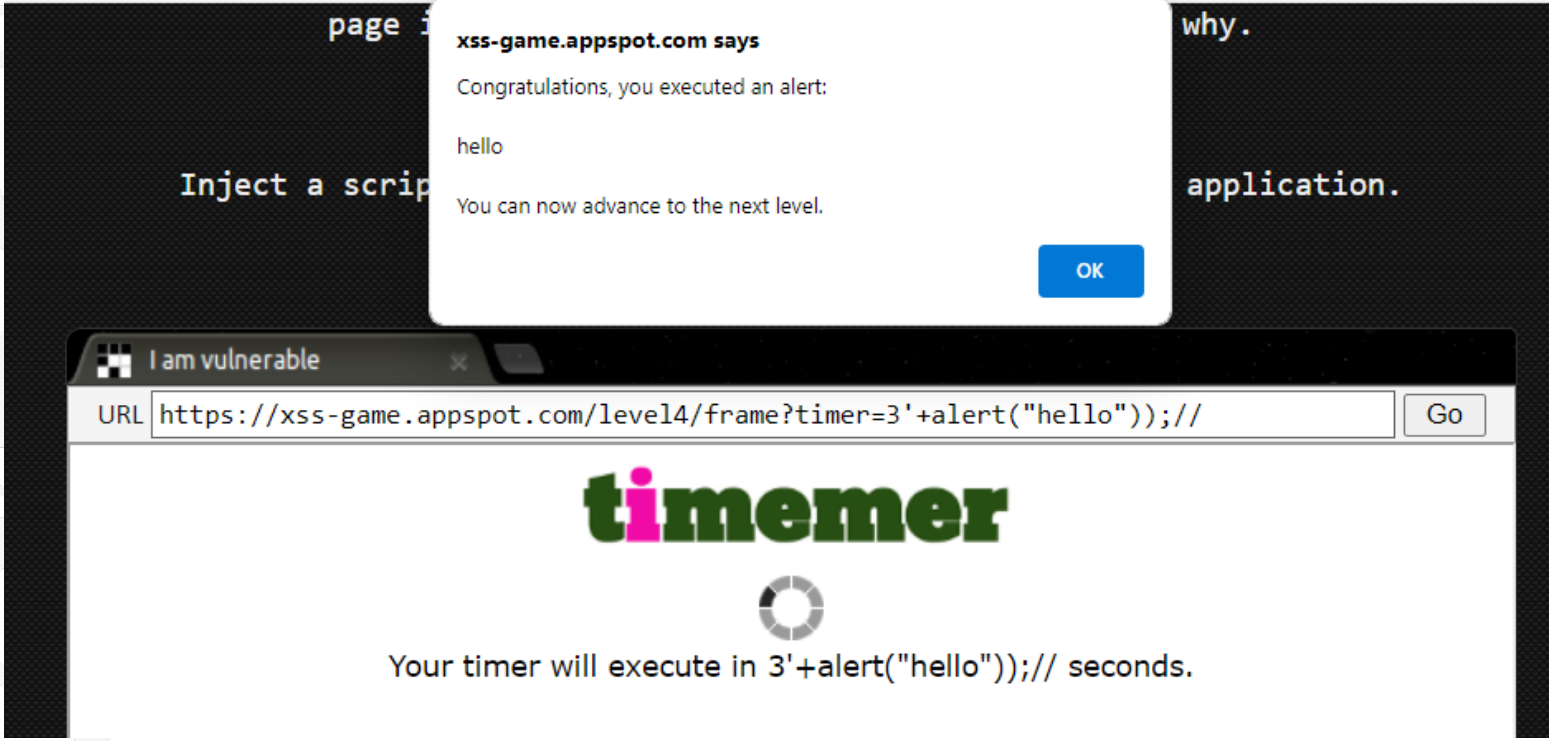
```
 8       <script>
 9         function startTimer(seconds) {
10           seconds = parseInt(seconds) || 3;
11           setTimeout(function() {
12             window.confirm("Time is up!");
13             window.history.back();
14           }, seconds * 1000);
15         }
16       </script>
17     </head>
18     <body id="level4">
19       <img src="/static/logos/level4.png" />
20       <br>
21       <img src="/static/loading.gif" onload="startTimer('{{ timer }}');" />
22       <br>
23       <div id="message">Your timer will execute in {{ timer }} seconds.</div>
24     </body>
25   </html>
```

potential execution sink

Performing the Attack



Performing the Attack - 2nd solution



Level 5

Mission Description

Cross-site scripting isn't *just* about correctly escaping data. Sometimes, attackers can do bad things even without injecting new elements into the DOM.

Mission Objective

Inject a script to pop up an `alert()` in the context of the application.

Your Target

Exploring the Website

Exploring the Website

Inspecting the Source Code

confirm.html | level.py | signup.html | welcome.html

```
1  <!doctype html>
2  <html>
3    <head>
4      <!-- Internal game scripts/styles, mostly boring stuff -->
5      <script src="/static/game-frame.js"></script>
6      <link rel="stylesheet" href="/static/game-frame-styles.css" />
7    </head>
8
9    <body id="level5">
10     Welcome! Today we are announcing the much anticipated<br><br>
11     <img src="/static/logos/level5.png" /><br><br>
12
13     <a href="/level5/frame/signup?next=confirm">Sign up</a>
14     for an exclusive Beta.
15   </body>
16 </html>
```

UNIVERSITY of York

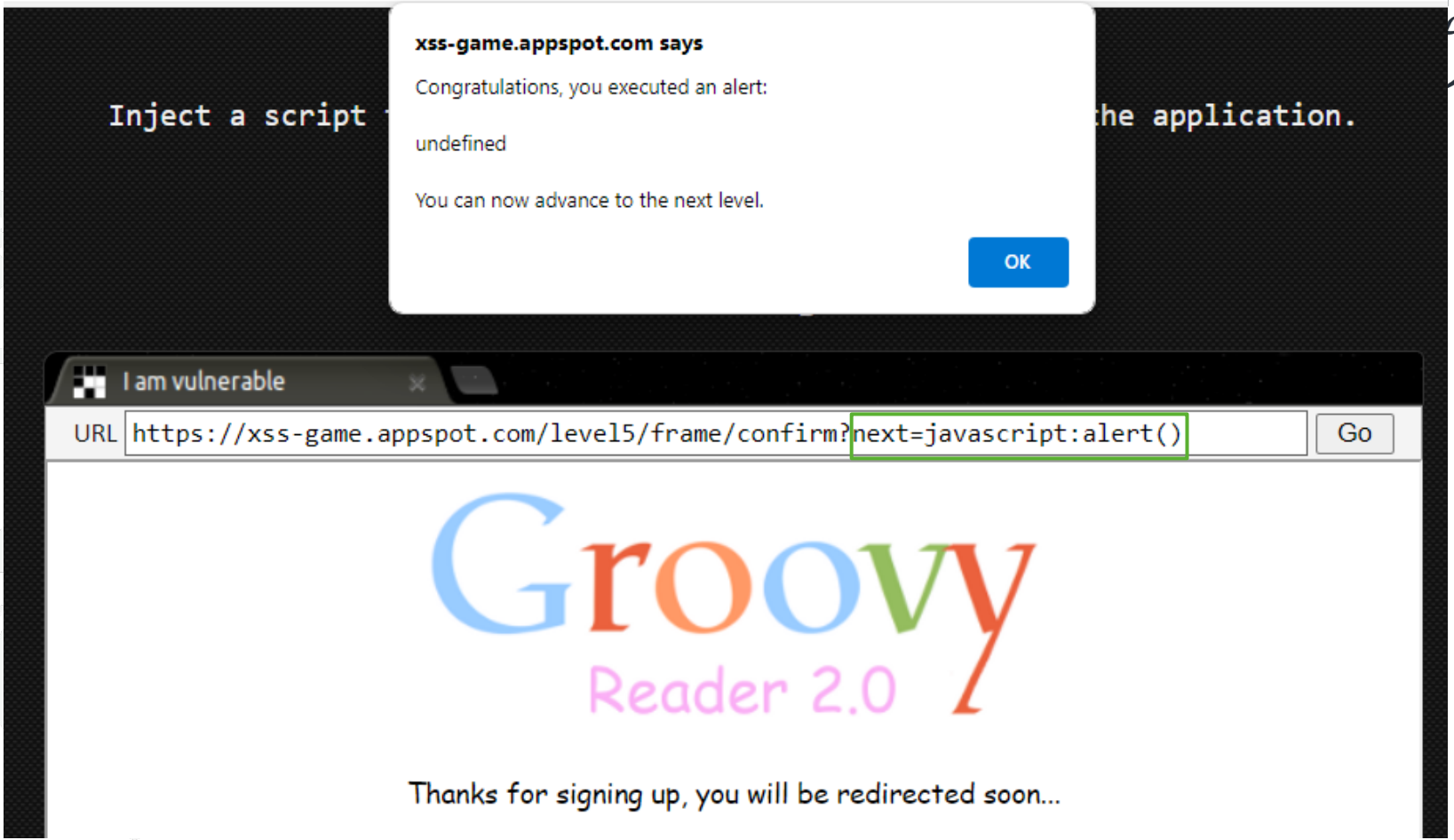
confirm.html | level.py | signup.html | welcome.html

```
<!doctype html>
<html>
  <head>
    <!-- Internal game scripts/styles, mostly boring stuff -->
    <script src="/static/game-frame.js"></script>
    <link rel="stylesheet" href="/static/game-frame-styles.css" />
  </head>

  <body id="level5">
    <img src="/static/logos/level5.png" /><br><br>
    <!-- We're ignoring the email, but the poor user will never know! -->
    Enter email: <input id="reader-email" name="email" value="">

    <br><br>
    <a href="{{ next }}">Next >></a>
  </body>
</html>
```

confirm.html | level.py | signup.html | welcome.html

```
1  <!doctype html>
2  <html>
3    <head>
4      <!-- Internal game scripts/styles, mostly boring stuff -->
5      <script src="/static/game-frame.js"></script>
6      <link rel="stylesheet" href="/static/game-frame-styles.css" />
7    </head>
8
9    <body id="level5">
10     <img src="/static/logos/level5.png" /><br><br>
11     Thanks for signing up, you will be redirected soon...
12     <script>
13       setTimeout(function() { window.location = '{{ next }}'; }, 5000);
14     </script>
15   </body>
16 </html>
```


Inject a script
the application.
xss-game.appspot.com says
Congratulations, you executed an alert:
Hello
You can now advance to the next level.
OK
I am vulnerable
URL
https://xss-game.appspot.com/level5/frame/signup?next=javascript:alert("Hello")
Go
Groovy
Reader 2.0
Enter email:
Next >>


xss-game.appspot.com says
Congratulations, you executed an alert:
undefined
You can now advance to the next level.
OK
Inject a script
the application.
I am vulnerable
URL
https://xss-game.appspot.com/level5/frame/confirm?next=javascript:alert()
Go
Groovy
Reader 2.0
Thanks for signing up, you will be redirected soon...

Level 6

Mission Description

Complex web applications sometimes have the capability to dynamically load JavaScript libraries based on the value of their URL parameters or part of **location.hash**.

This is very tricky to get right -- allowing user input to influence the URL when loading scripts or other potentially dangerous types of data such as **XMLHttpRequest** often leads to serious vulnerabilities.

Mission Objective

Find a way to make the application request an external file which will cause it to execute an **alert()**.

Your Target

Inspecting the Source Code

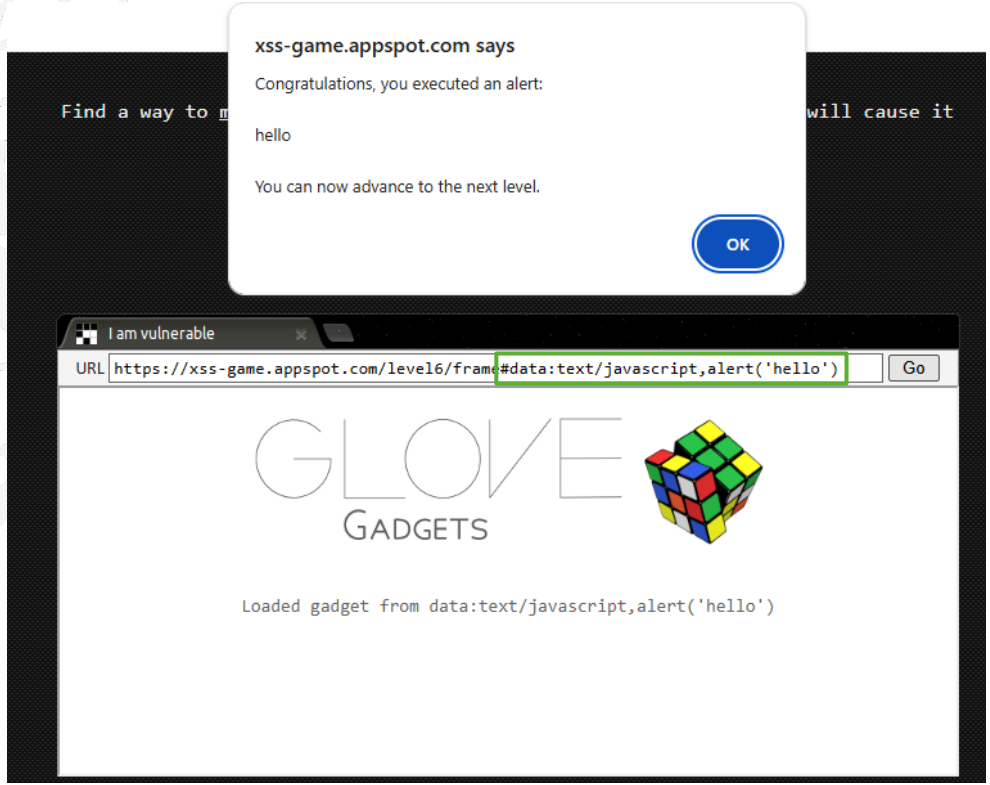
Some escaping for explicit loading external files

```
17      function includeGadget(url) {
18        var scriptEl = document.createElement('script');
19
20        // This will totally prevent us from loading evil URLs!
21        if (url.match(/^https?:\/\//)) {
22          setInnerText(document.getElementById("log"),
23            "Sorry, cannot load a URL containing \"http\".");
24          return;
25        }
26
27        // Load this awesome gadget
28        scriptEl.src = url;
29
30        // Show log messages
31        scriptEl.onload = function() {
32          setInnerText(document.getElementById("log"),
33            "Loaded gadget from " + url);
34        }
```

Trying to use HTTP

explicit loading of external file has been escaped

Performing the Attack



Using data URI scheme

Further Reading

[1] https://xss-game.appspot.com/

[2] OWASP, Cross-Site-Scripting (XSS) https://owasp.org/www-community/attacks/xss/

[3] JavaScript Tutorial, https://www.w3schools.com/js/