Secure From Scratch

With security in mind,
From the first line of code

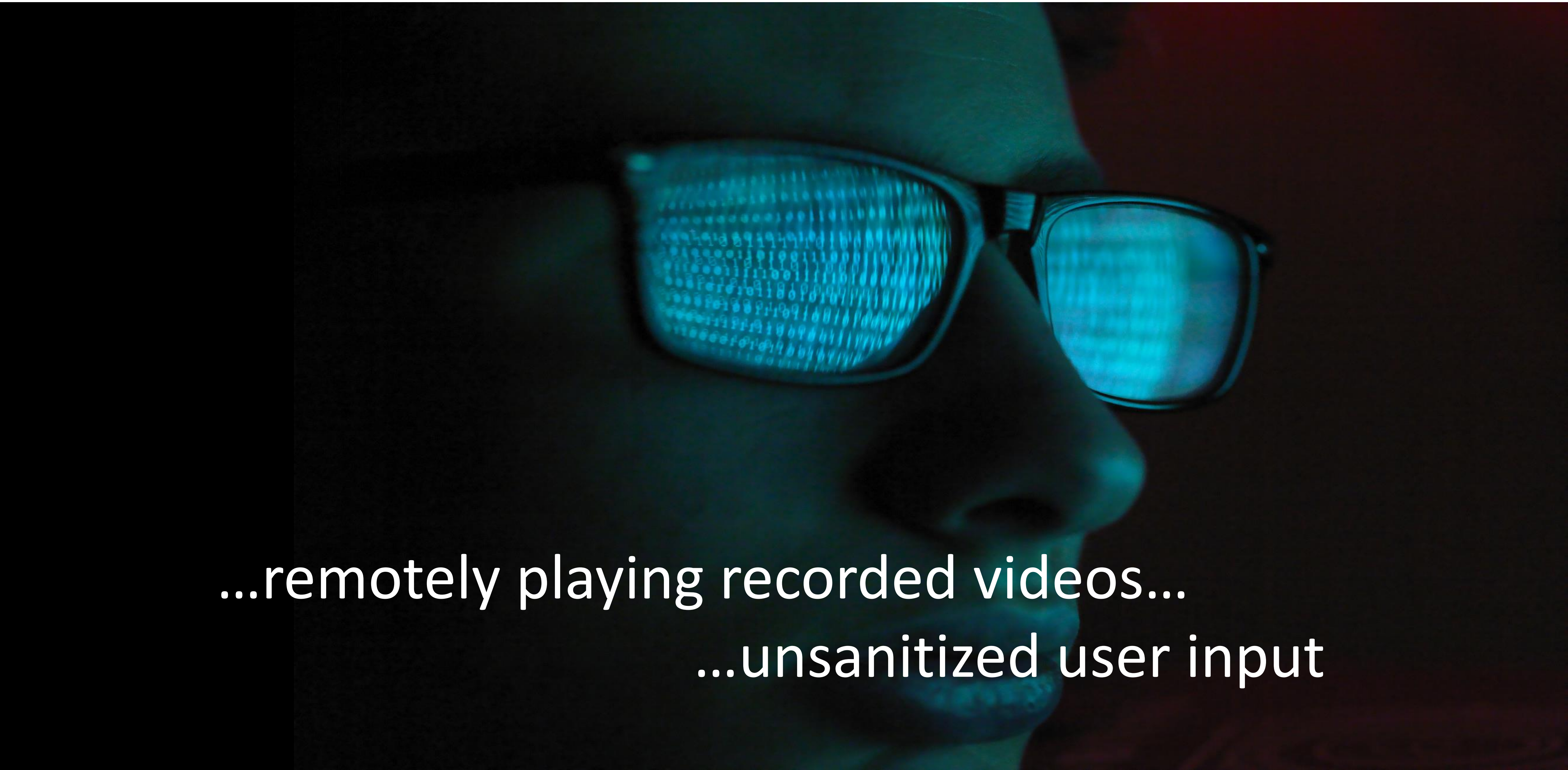
After this workshop

Write code with less security vulnerabilities

Secure From Scratch

Me

Secure From Scratch



…remotely playing recorded videos…

…unsanitized user input

Secure From Scratch

Secure From Scratch

Demo

Secure From Scratch

```
string fn = GetVar(sRequest, "fn");
...

case "playfile": {
try {
    string subdir = Helper.GetDirectory(otid, oid);
    string filename = Helper.GetMediaDirectory(otid, oid);
     ...
    filename += subdir + @"\" + fn;

    try
    {
        Process.Start(filename);
```

Request URL

Yariv Tal

University lecturer

Bootcamps mentor

Seasoned developer (WhyT software)

Application security researcher

Drug of choice: Travelling & Roller coasters

Secure From Scratch

Or Sahar

Senior security researcher

Secure code Instructor

Application security consultation and PT

A veteran developer

Drug of choice : CVEs & Snowy mountains

Getting to Know You

- Backend/Front end?
- Java/C#/Python/C++/NodeJS/Typescript/Other?
- Experience?
- Security knowledge?
- i++ or ++i?

What Is This Workshop About?

This is a Secure Programming Workshop

It’s a **mindset**

Can be applied to any programming language

Programming, nothing more.

Not SDLC, Secure By design, Threat modeling etc.

What is Secure Programming?

Programming that leads to *less* Security Bugs

Secure coding is … developing computer software in such a way that guards against the accidental introduction of *security vulnerabilities*.

Defects, bugs and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities.
… most vulnerabilities stem ***from a relatively small number of common software programming errors***.

Wikipedia

What is a Security Bug?

A bug that leads to a security vulnerability.

What is a Security Vulnerability?

So many (incomplete) definitions!

In OWASP We Trust:

A vulnerability is a hole or a weakness in the application .. that **allows an attacker to cause harm** to the stakeholders of an application.

Stakeholders include the application owner, application users, and other entities that rely on the application.

The Goal of Secure Programming

Less Security Bugs

→ Less Vulnerabilities

→ Less Exploitations

→ Less Harm

The Goal of Secure From Scratch

Involve in the developers’ careers ASAP

- Make Secure Programming the default
- Form good habits

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

Trust Boundaries

Workshop Outline

Part 1:
The Need

Cost of Vulnerabilities

A lot!

(Out of scope)

Part 2:
Getting To Know The Exercise Environment

Case In-Point – Tasks Server App

Collaboratively manage tasks

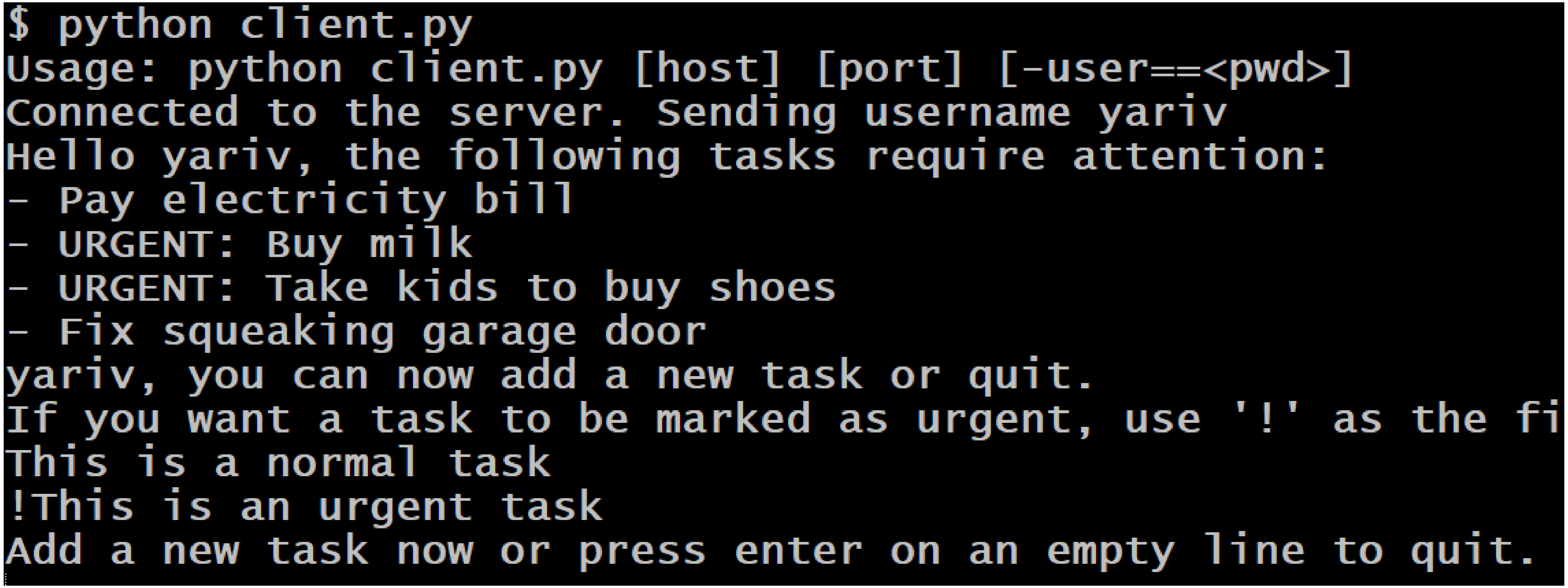
Text-based

Central server stores tasks

Simple client to access

User-based

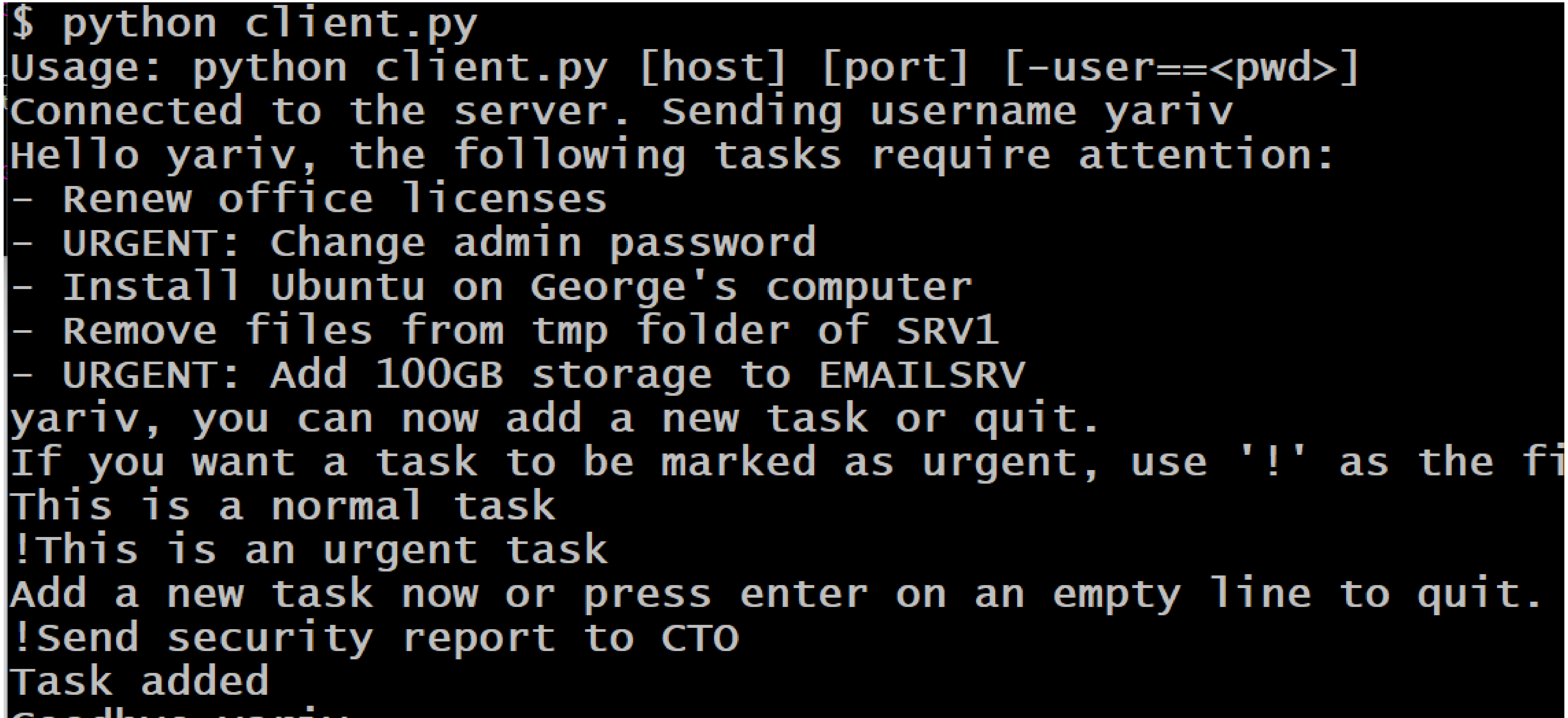
Case In-Point – Tasks Server App (Personal usage)



```
$ python client.py
Usage: python client.py [host] [port] [-user==<pwd>]
Connected to the server. Sending username yariv
Hello yariv, the following tasks require attention:
- Pay electricity bill
- URGENT: Buy milk
- URGENT: Take kids to buy shoes
- Fix squeaking garage door
yariv, you can now add a new task or quit.
If you want a task to be marked as urgent, use '!' as the fi
This is a normal task
!This is an urgent task
Add a new task now or press enter on an empty line to quit.
```

Secure From Scratch

Case In-Point – Tasks Server App (IT usage)



```
$ python client.py
Usage: python client.py [host] [port] [-user==<pwd>]
Connected to the server. Sending username yariv
Hello yariv, the following tasks require attention:
- Renew office licenses
- URGENT: Change admin password
- Install Ubuntu on George's computer
- Remove files from tmp folder of SRV1
- URGENT: Add 100GB storage to EMAILSRV
yariv, you can now add a new task or quit.
If you want a task to be marked as urgent, use '!' as the fi
This is a normal task
!This is an urgent task
Add a new task now or press enter on an empty line to quit.
!Send security report to CTO
Task added
```

Text Based? Console? What?

This isn't the 80's!

The world is in the Web!

Remember Your Goal: Learning

How did you learn programming?

What programs did you write?

Why?

Remember Your Goal: Learning

Learning is easier when:

- Less overhead
- Problems are reduced to their essence
- Principles remain the same

Secure From Scratch

Text Based = Same problems, but Simpler

Hacking did NOT start in the 2000’s

Not even the 90’s

Secure From Scratch

Secure From Scratch

Text Based = Same problems, but Simpler

- Easier to debug
- Easier to test
- Simpler to understand flow
- Same problems!

LAB: Setting up the environment

5 min

1. Open a command line / shell / terminal
2. Create an exercises directory, i.e. sfs: mkdir sfs
3. Change directory into sfs: cd sfs
4. Create a directory within called part2: mkdir part2
5. Change directory into part2: cd part2
6. Clone the source code from Github: git clone https://github.com/SecureFromScratch/HalfDay2023/

LAB: Get to know the Tasks Server app

8 min

1. Choose your programming language.
2. Read language_options.txt
3. SERVER
   a) Folder: “start_here”
   b) Build & Execute
   c) Look at tasks.txt
   d) Look at tasks.log

   CLIENT
   a) Folder: “client”
   b) Build & Execute
   c) Add a task
   d) Add an urgent task
   e) What user is sent?

Tasks Server App – Code Overview

<<<CODE OVERVIEW>>>

Part 3:
Adding Authorization

Problem: Anyone can assign urgent tasks

Let’s limit it to only one user: “theboss”

Discussion: How would we implement this?

LAB: Add Urgent Task Creation Restriction

10 min

1. Add a limit so only “theboss” can create urgent tasks
2. Test the feature
   1. Normal users should not be able to create urgent tasks
   2. theboss should be able to create an urgent task
3. The client has a –user= flag to facilitate testing
4. You can use –user=shutdown to cause server to shutdown

https://github.com/SecureFromScratch/HalfDay2023/

Tasks Server App – Authorizing “theboss”

<<<CODE OVERVIEW>>>

Complete the Sentence: Quick and …

```
if new_task_description:
  if new_task_description.startswith("!")
        and username != "theboss":
    connection.writeln("You are not authorized”)
  else:
    tasks_mgr.add(username, new_task_description)
    connection.writeln("Task added")
```

PREVENT - SFS Principles

$\boxed{\text{Priority – Security is the first priority}}$ → Guides us

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

Trust Boundaries

DISCUSSION: What’s Wrong with Quick and Dirty?

```
if new_task_description:
  if new_task_description.startswith("!")
        and username != "theboss":
    connection.writeln("You are not authorized”)
  else:
    tasks_mgr.add(username, new_task_description)
    connection.writeln("Task added")
```

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

Trust Boundaries

X ~~*Neat Code*~~: Should not have hardcoded data

```
if (newTaskDescription.startsWith("!")
                 && !a_username.equals("theboss")) {
    a_connection.writeln("You are not authorized"));
}
else {
    a_tasksMgr.Add(a_username, newTaskDescription);
    a_connection.writeln("Task added");
}
```

X ~~*Trust Boundaries*~~: Code vs. Source Code Storage

```
if (newTaskDescription.startsWith("!")
                        && !a_username.equals("theboss")) {
      a_connection.writeln("You are not authorized"));
}
else {
      a_tasksMgr.Add(a_username, newTaskDescription);
      a_connection.writeln("Task added");
}
```

X ~~*Neat Code*~~: Violates Single responsibility principle

```
if (newTaskDescription.startsWith("!")
                && !a_username.equals("theboss")) {
    a_connection.writeln("You are not authorized"));
}
else {
    a_tasksMgr.Add(a_username, newTaskDescription);
    a_connection.writeln("Task added");
}
```

Secure From Scratch

X ~~*Neat Code*~~: Don’t Repeat Yourself (DRY)

```
if (newTaskDescription.startsWith("!")
                && !a_username.equals("theboss")) {
    a_connection.writeln("You are not authorized"));
}
else {
    a_tasksMgr.Add(a_username, newTaskDescription);
    a_connection.writeln("Task added");
}
```

Secure From Scratch

X ~~*Easy to Use Safely*~~: easy to forget auth check

```
if new_task_description:
  if new_task_description.startswith("!")
        and username != "theboss":
    connection.writeln("You are not authorized”)
  else:
    tasks_mgr.add(username, new_task_description)
    connection.writeln("Task added")
```

Authorizing “theboss” – What is a Good way?

What would be the properties of a good way?

Ensures authorization check

Centralized code

DRY

No hard coded user details

How Can we Ensure Authorization Check?

```
if new_task_description:
  if new_task_description.startswith("!")
        and username != "theboss":
    connection.writeln("You are not authorized”)
  else:
    tasks_mgr.add(username, new_task_description)
    connection.writeln("Task added")
```

How Can we Ensure Authorization Check?

```
~~if new_task_description:~~
~~  if new_task_description.startswith("!")~~
~~          and username != "theboss":~~
~~    connection.writeln("You are not authorized”)~~
~~  else:~~
    tasks_mgr.add(authorization,new_task_description)
    connection.writeln("Task added")
```

Authorization Class

Authorization class creation and methods:

```
Authorization authorization =
     AuthMgr.getAuthorization(username, s_logger);

public class Authorization {
    public String getUsername() ...
    public boolean allows(String a_right) ...
    public void throwIfNotAllowed(String a_right)
                                      throws InvalidAuth
...
```

LAB: Setting up the environment

3 min

1. Open a command line / shell / terminal
2. Change directory into your exercises directory
3. Create a directory within called part3: mkdir part3
4. Change directory into part3: cd part3
5. Clone the source code from Github: git clone https://github.com/SecureFromScratch/HalfDay2023/
6. Switch branches: git switch improve_auth

AuthMgr.getAuthorization(…)

<<<CODE OVERVIEW>>>

Code is on a git branch.

Use:

```
git switch improve_auth
```

Secure From Scratch

auth.txt

theboss:urgenttask

:viewactive

What are we missing?

Using Authorization Class

ALL Actions Must Be Checked for Authorization:

```
public class TasksManager {
    ...
    public Task[] GetActiveTasks(
                          Authorization a_authorization)
                          throws InvalidAuth {
        a_authorization.throwIfNotAllowed(
                               AuthMgr.VIEW_ACTIVE);
        try {
            ...
```

Using Authorization Class

Handle *InvalidAuth* Exception:

```
public class TasksManager {
    ...
    public Task[] GetActiveTasks(
                                Authorization a_authorization)
                                throws InvalidAuth {
        a_authorization.throwIfNotAllowed(
                                        AuthMgr.VIEW_ACTIVE);
        try {
            ...
```

PREVENT - SFS Principles

Priority – Security is the first priority → Guides us

Reporting & logging

Easy to use safely

Verify

$\boxed{\text{Errors \& exceptions}}$

Neat code

Trust Boundaries

LAB: Fix Missing Authorization Code

25 min

1. Fix broken/missing code on “improve_auth” branch:

```
Authorization authorization =
        AuthMgr.getAuthorization(username, s_logger);

class Autherization {
    String getUsername() ...
    boolean allows(String a_right) ...
    void throwIfNotAllowed(String a_right)
                                    throws InvalidAuth
```

git switch improve_auth

Fix Missing Authorization Code

<<<CODE OVERVIEW>>>

Secure From Scratch

Take Aways

Beware quick solutions

Centralize security related code (Authorization)

Keep security related data away from code

Avoid half measures

Part 4:
Breaking It

LAB: Learning How The Enemy Thinks

10 min

1. We are going to do some social engineering
2. We want to add the following task:
   Create an system account for “hack” with pwd “1234”
3. We want IT to believe it is from the boss
   (so they will perform the task without questions)
4. HINT: Who is the only one that can create urgent tasks?

Is this realistic?

your own code *-or-* git switch good_auth

Urgent Task from Boss – Exploitation 1

Change the tasks.txt file using notepad

Change the auth.txt file using notepad

Not fair?

Question: Who’s responsible to prevent this?

<<DISCUSSION>>

Secure From Scratch

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

$\boxed{\textbf{T}\text{rust Boundaries}}$ → Data flows between File & Server App

Secure From Scratch

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

$\boxed{\textbf{T}\text{rust Boundaries}}$ → Data flows between ~~File~~ DB & Server App

Urgent Task from Boss – Exploitation 2

What happens if:

- Add a *normal* task,
- Prefix it with “URGENT: “

Not fair?

Urgent Task from Boss – Exploitation 2

Problem: Urgent tasks can be visually faked

Discussion: Solution Ideas?

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

$\boxed{\textbf{T}\text{rust Boundaries}}$ → Data flows between User & Server App

Input Validation

Active measure for Input→Code Trust Boundary

User supplied input could be malicious

Input Validation attempts to ensure it is not

LAB: Setting up the environment

3 min

1. Open a command line / shell / terminal
2. Change directory into your exercises directory
3. Create a directory within called part4: mkdir part4
4. Change directory into part4: cd part4
5. Clone the source code from Github: git clone https://github.com/SecureFromScratch/HalfDay2023/
6. Switch branches: git switch good_auth

5 min

LAB: Prevent prefixing with URGENT:

1. Add input validation for tasks
2. Ensure a task does not begin with URGENT:

https://github.com/SecureFromScratch/HalfDay2023/

your own code *-or-* git switch good_auth

Prevent prefixing with URGENT:

<<<CODE OVERVIEW>>>

5 min

LAB: Bypass the Input Validation

1. We still want to add the following task:
   URGENT: Create account “hacker1” with pwd “123456”
2. Our new Input Validation blocks it
3. Or does it?
   Find a *good enough* way to bypass the blocking

your own code

Bypassing URGENT: Prefix Prevention

Humans don’t have good eyesight…

We can use that.

 URGENT:

URGENT;

URGENT :

** Better chance of working if the only active urgent task

Prevent bypassing input validation

<<<Discussion>>>

The Problem is Blocking

Our validation checks what to BLOCK.

Almost all attempts to get it right are doomed.

The Problem is Blocking

There is always
something
you did not think about.

A Different Approach

- Change the internal design
- Change the visuals to be unambiguous

```
Urgent? Task
NO      Renew office licenses
YES     Change admin password
NO      Install Ubuntu on George's computer
NO      Remove files from tmp folder of SRV1
YES     Add 100GB storage to EMAILSRV
```

LAB: Change Visual Display of Tasks

5 min

1. Change how tasks are stored
2. Change how tasks are displayed

```
Urgent? Task
NO      Renew office licenses
YES     Change admin password
NO      Install Ubuntu on George's computer
```

https://github.com/SecureFromScratch/HalfDay2023/

your own code *-or-* git switch good_auth

No Input Validation? Can That Work?

Previous solution has no input validation.

Can we bypass this design?

Can we break it?

No Input Validation → Someone will find a way

The Road to Exploitation: Strings

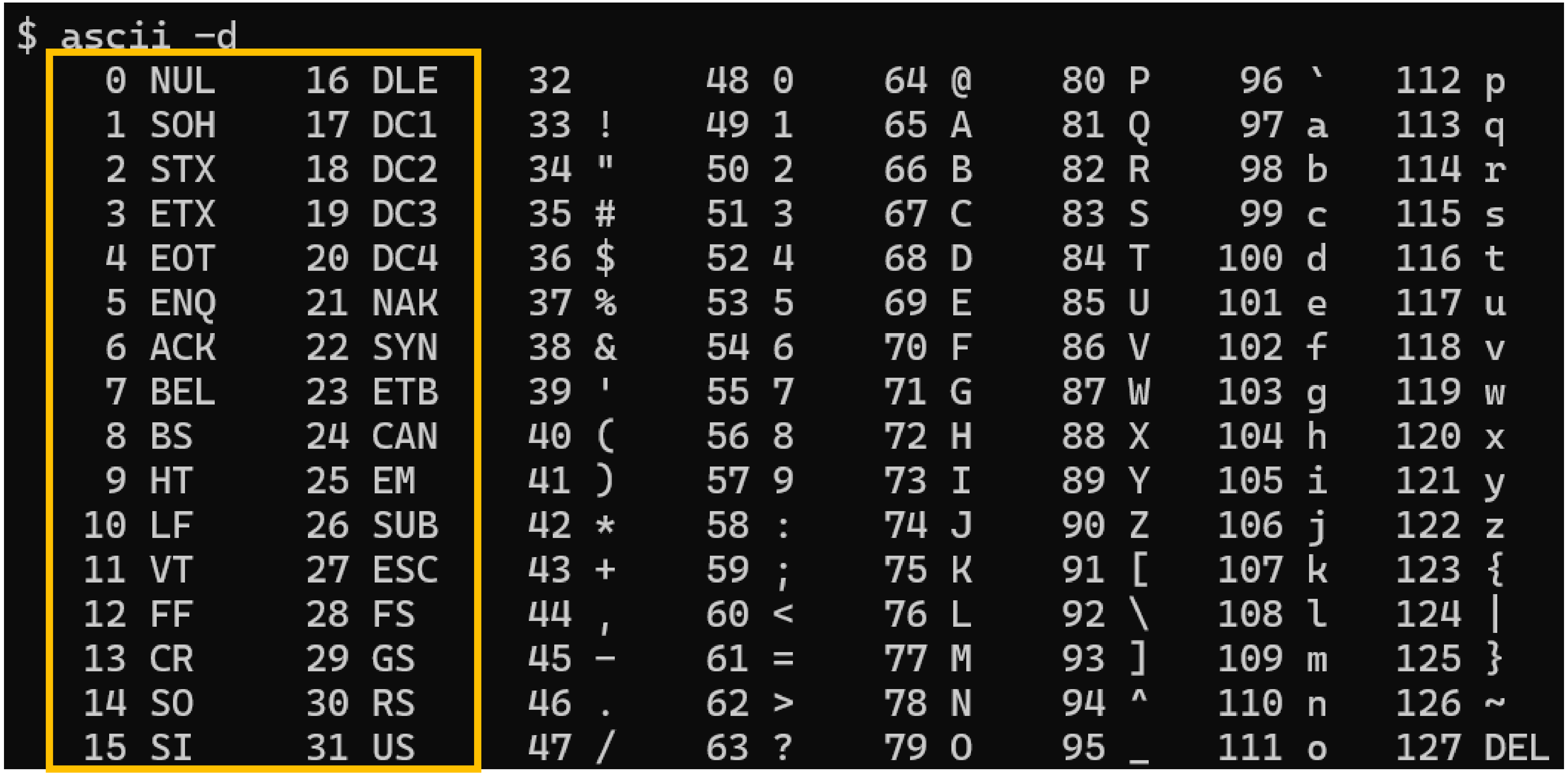
Characters and Letters are not the same thing!

You might think that strings hold letters, but they don’t

(Re)introducing the ASCII table:

Secure From Scratch

The ASCII Table



```
$ ascii -d
    0 NUL    16 DLE    32      48 0    64 @    80 P    96 `   112 p
    1 SOH    17 DC1    33 !    49 1    65 A    81 Q    97 a   113 q
    2 STX    18 DC2    34 "    50 2    66 B    82 R    98 b   114 r
    3 ETX    19 DC3    35 #    51 3    67 C    83 S    99 c   115 s
    4 EOT    20 DC4    36 $    52 4    68 D    84 T   100 d   116 t
    5 ENQ    21 NAK    37 %    53 5    69 E    85 U   101 e   117 u
    6 ACK    22 SYN    38 &    54 6    70 F    86 V   102 f   118 v
    7 BEL    23 ETB    39 '    55 7    71 G    87 W   103 g   119 w
    8 BS     24 CAN    40 (    56 8    72 H    88 X   104 h   120 x
    9 HT     25 EM     41 )    57 9    73 I    89 Y   105 i   121 y
   10 LF     26 SUB    42 *    58 :    74 J    90 Z   106 j   122 z
   11 VT     27 ESC    43 +    59 ;    75 K    91 [   107 k   123 {
   12 FF     28 FS     44 ,    60 <    76 L    92 \   108 l   124 |
   13 CR     29 GS     45 -    61 =    77 M    93 ]   109 m   125 }
   14 SO     30 RS     46 .    62 >    78 N    94 ^   110 n   126 ~
   15 SI     31 US     47 /    63 ?    79 O    95 _   111 o   127 DEL
```

Secure From Scratch

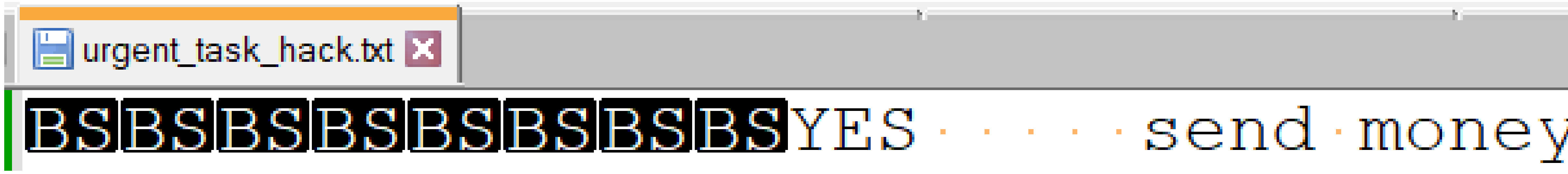
The Road to Exploitation: Control Characters

How do I type Ctrl Characters?

Use a Hex Editor

echo –e "\x08\x08\x08\x08\x08\x08\x08\x08YES   Send…"

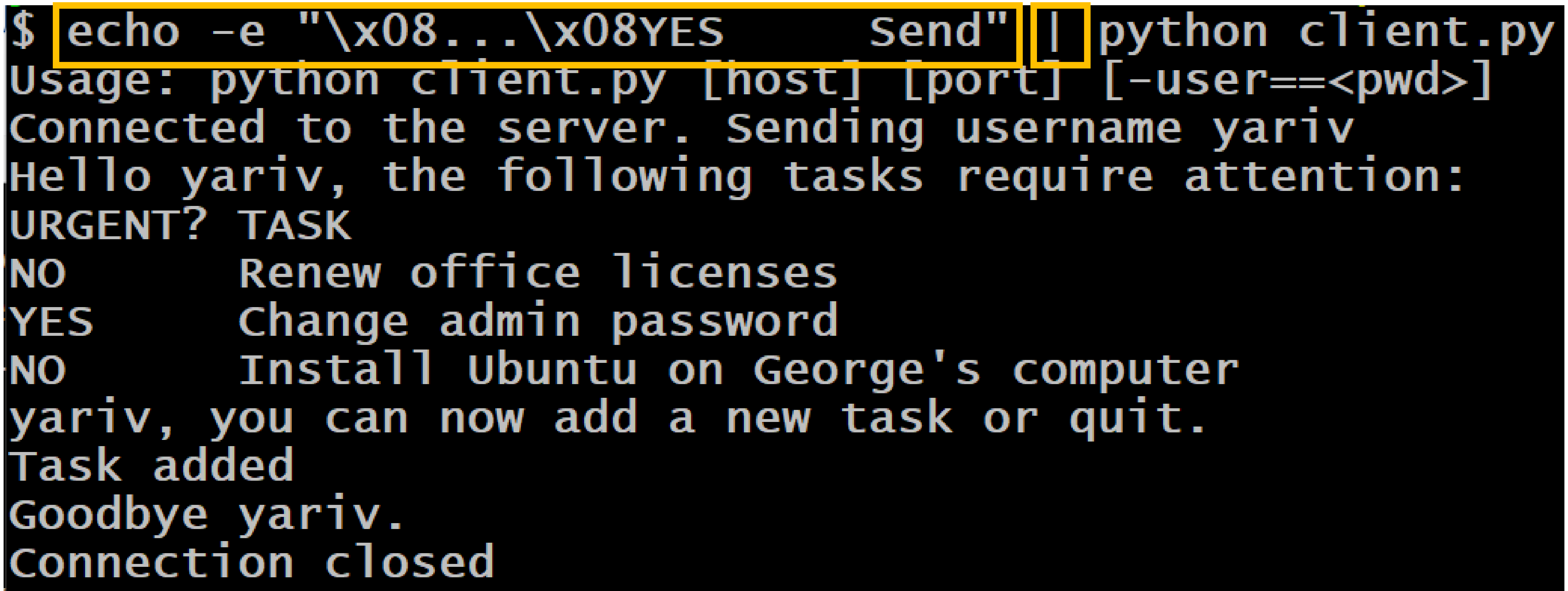
Write a program that outputs them



```
BSBSBSBSBSBSBSBSYES·····send·money
```

How To: Input Ctrl Characters in Client

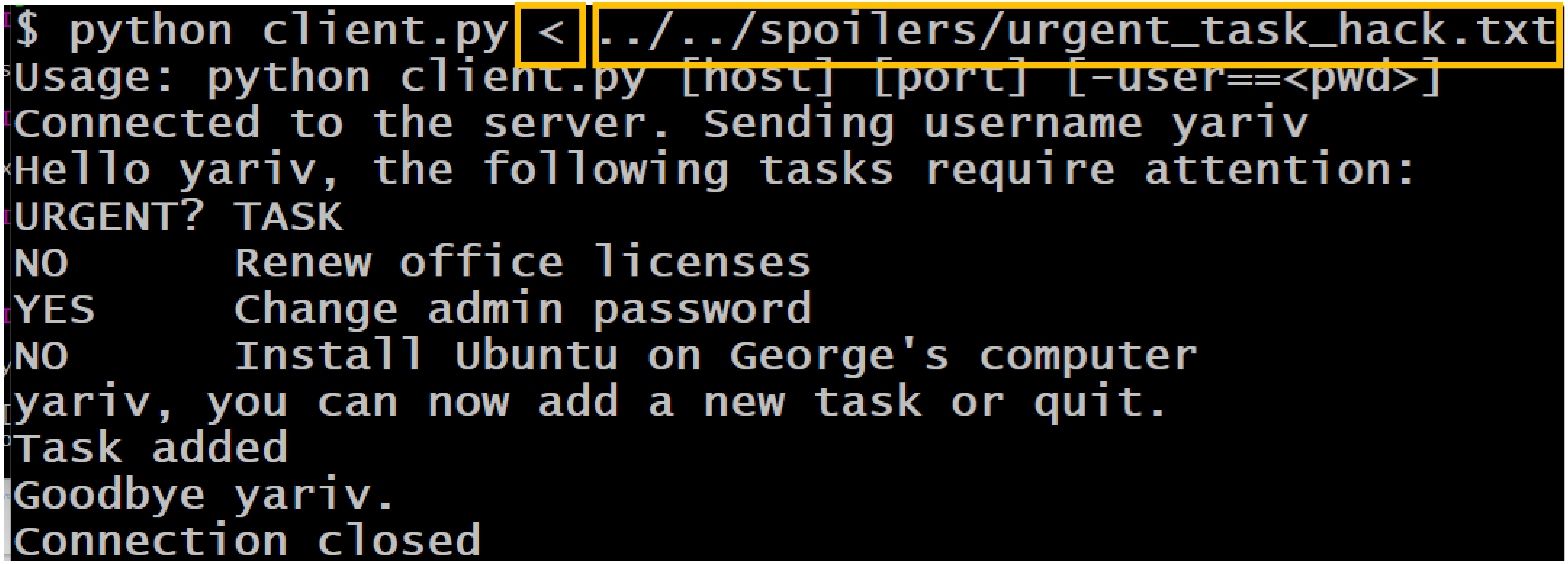
Use echo + pipe:



```
$ echo -e "\x08...\x08YES        Send" | python client.py
Usage: python client.py [host] [port] [-user==<pwd>]
Connected to the server. Sending username yariv
Hello yariv, the following tasks require attention:
URGENT? TASK
NO      Renew office licenses
YES     Change admin password
NO      Install Ubuntu on George's computer
yariv, you can now add a new task or quit.
Task added
Goodbye yariv.
Connection closed
```

How To: Input Ctrl Characters in Client

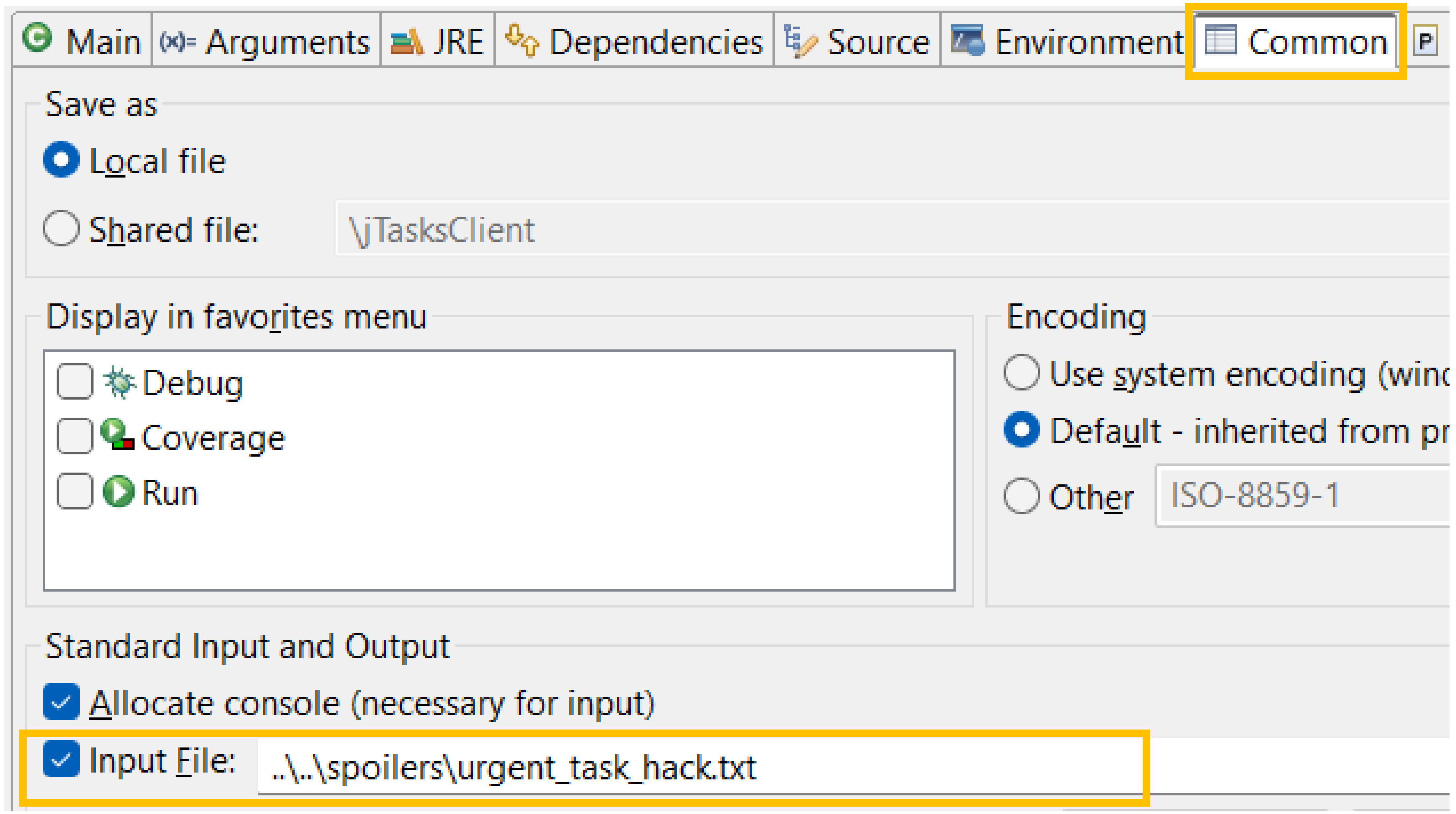
Use input redirection (shell):



```
$ python client.py < ../../spoilers/urgent_task_hack.txt
Usage: python client.py [host] [port] [-user==<pwd>]
Connected to the server. Sending username yariv
Hello yariv, the following tasks require attention:
URGENT? TASK
NO      Renew office licenses
YES     Change admin password
NO      Install Ubuntu on George's computer
yariv, you can now add a new task or quit.
Task added
Goodbye yariv.
Connection closed
```

How To: Input Ctrl Characters in Client

Use input redirection (eclipse):



Save as
[✓ Local file]
Shared file: \jTasksClient

Display in favorites menu
☐ Debug
☐ Coverage
☐ Run

Encoding
Use system encoding (winc
[✓ Default - inherited from pr]
Other ISO-8859-1

Standard Input and Output
☑ Allocate console (necessary for input)
☑ Input File: ..\..\spoilers\urgent_task_hack.txt

LAB: Setting up the environment

3 min

1. Open a command line / shell / terminal
2. Change directory into your exercises directory
3. Create a directory within called ctrl: mkdir ctrl
4. Change directory into ctrl: cd ctrl
5. Clone the source code from Github: git clone https://github.com/SecureFromScratch/HalfDay2023/
6. Switch branches: git switch columns

LAB: Exploit Ctrl Characters

5 min

1. Prepare a task that contains ctrl-characters
   Or you can use ../../spoilers/urgent_task_hack.txt
2. Execute the client with the prepared task
3. You can try with echo + pipe vs. input redirection:

```
echo –e “\x08\x08\x08\x08\x08\x08\x08\x08YES     Send…” | python client.py
```

or

```
python client.py < ../../spoilers/urgent_task_hack.txt
```

LAB Conclusions

User input shouldn't be trusted

User input is dangerous.

User Input → Bad

LAB Conclusions

Unfortunately, we need user input.

We need a solution.

Introducing: Welcome (Allow) Lists

The reverse of blocking.

We say what we allow.

What do we allow here?

Secure From Scratch

PREVENT - SFS Principles

$\boxed{\text{Priority – Security is the first priority}}$ → Can’t be lazy…

Reporting & logging

$\boxed{\text{Easy to use safely}}$

Verify

Errors & exceptions

Neat code

Trust Boundaries

Secure From Scratch

Take Aways

Flow between Trust Boundaries requires Active Measures

Input must pass Validation

Validation *always* requires a Welcome (Allow) List

Part 5:
Building Block: TextLine

We No Longer Need Control Characters

They cause so many bugs

But, they are still supported by every language:

String

So, let’s get rid of them

How Would We Get Rid of Control Characters?

<<DISCUSS>>

Control Characters – The Solution

GOAL: Automatically Block Control Characters

SOLUTION: Replace class String

class TextLine:

- Will block any control character
- Validation: Notify with an error
- Sanitization: Erase control characters

LAB: TextLine class (partial) implementation

15 min

1. Write a TextLine class
   - It should have a BlockCtrlChars(str) method
2. See that you block control characters
3. Create an application that uses TextLine class
4. Test your code

TextLine

<<CODE REVIEW>>

TextLine – What Do We Get From It?

Priority – Security is the first priority

Reporting & logging

$\boxed{\textbf{E}\text{asy to use safely}}$

Verify

Errors & exceptions

Nothing is new

Trust Boundaries

Easy to Use Safely

- Don’t rely on your memory
  - Don’t require writing additional safety checks

- Change *how* you code
  - Safety should be inherent in your writing

~~String~~ → TextLine

Feedback Form: https://tinyurl.com/mrhav3pj

Part 6:
Logging

Why Programs Output Their State

Know what the program is doing right now

Know what the program did in the past

Security for example:

- Detection brute force attacks
- Detection of broken access control
- Identifying potential vulnerabilities
- Forensics

Forensics

Log is the Mirror of the Breach

- Identify extent of damage
- Identify vulnerability used in breach
- Identify hacker identity

Why use a logger?

- Easy to use
- Customizations
    - Severity
    - File system / Database / OS event Log

Have you noticed tasks.log?

Created by logging

What can we use it for?

Logging is Easy! Hello Secure World

```java
import java.util.logging.Level;
import java.util.logging.Logger;

public final class HelloSecureWorld {
    public static void main(final String[] args) {
        final Logger logger = Logger.getLogger("main");
        logger.log(Level.INFO, "Hello Secure World!");
    }
}
```

LAB: Check out Hello Secure World

3 min

1. Find the folder for Hello Secure World:
   It should be in the directory part2 (main branch)
2. Execute the program
3. Check out the log

Logging is the Mirror of the Breach

We’ll do a small simulation to understand what this means.

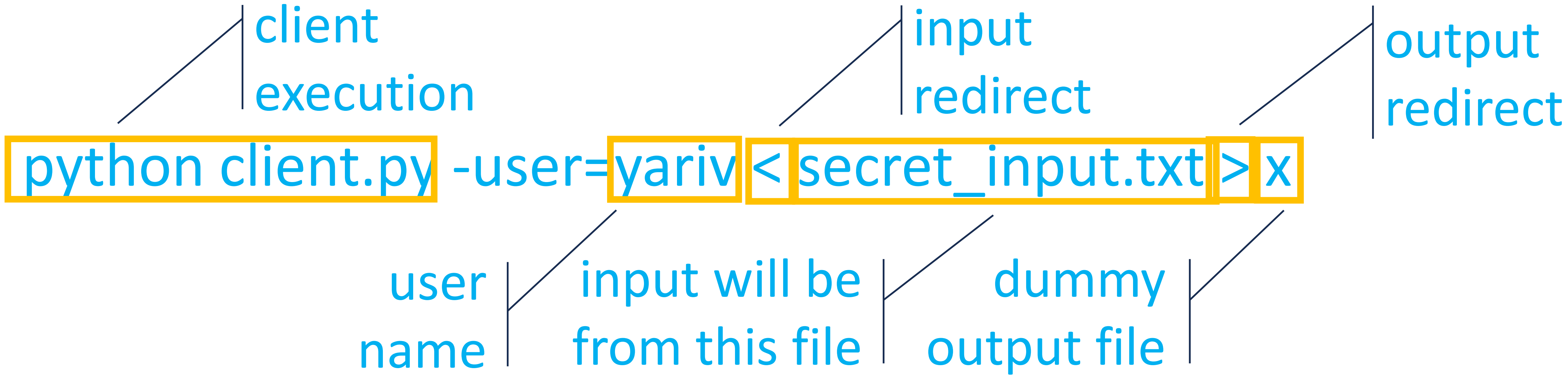
Logging is the Mirror of the Breach

We will simulate:

- 3 users using the client
- Each will try to add 2 tasks
- One of them is trying to add unauthorized urgent tasks
- The malicious user should remain unknown to us!

Simulating a User Using The Client

We will use an execution line similar to the one below:



```
python client.py -user=yariv < secret_input.txt > x
```

LAB: Setting up the environment

3 min

1. Open a command line / shell / terminal
2. Change directory into your exercises directory
3. Create a directory within called sim: mkdir sim
4. Change directory into sim: cd sim
5. Clone the source code from Github: git clone https://github.com/SecureFromScratch/HalfDay2023/
6. Switch branches: git switch columns

LAB: Simulating Multiple Users

5 min

1. Use something akin to the lines below:

```
python client.py -user=gil < ../../spoilers/input_task1.txt > x
python client.py -user=ron < ../../spoilers/input_task2.txt > x
python client.py -user=jim < ../../spoilers/input_task3.txt > x
python client.py -user=ron < ../../spoilers/input_task4.txt > x
python client.py -user=jim < ../../spoilers/input_task5.txt > x
python client.py -user=gil < ../../spoilers/input_task6.txt > x
```

2. Can you identify a malicious user?

Identifying a Malicious User

<<DISCUSSION>>

LAB: Add Missing Logging Information

3 min

1. Add logging to the tasks server
   1. Add as much logging as you see fit
   2. Output as much information as you think could be useful
2. Rerun the previous multiple-users simulation
3. Can you identify who is the malicious user?

Add Missing Logging Information

<<DISCUSSION>>

How Did You Add Logging?

```
log.info(USERNAME + “ something”);
…
log.warning(USERNAME + “ something”);
…
log.info(USERNAME + “ something”);
…
```

Is this easy to use safely?

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

$\boxed{\textbf{N}\text{eat code}}$ → DRY/WET

Trust Boundaries

DRY – Don’t Repeat Yourself

aka Don’t copy-paste

EVER!

Implementing DRY for Logging Extras

```
import logging as baselogging

_usernameHolder = _UsernameHolder()
baselogging.basicConfig(filename=‘tasks.log’,filemode='a',
  format='%(asctime)s ... %(user)s %(message)s',
  datefmt='%H:%M:%S', level=baselogging.DEBUG)

def getLogger(name):
    baselogger = baselogging.getLogger(name)
    return baselogging.LoggerAdapter(
                  baselogger, {'user': _usernameHolder})
```

Implementing DRY for Logging Extras

```
from extralogging import logging
…
with UsernameScope(username):
  logger.info(f"logged in") # user added automatically
…
  try:
    display_active_tasks(tasks_mgr, authorization, c)
    perform_add_task_dialog(tasks_mgr, authorization, c)
  except authmgr.InvalidAuth as e:
    logger.warning(“…unauthorized operation…")
```

Implementing DRY for Logging Extras

```
from extralogging import logging
…
with UsernameScope(username):
  logger.info(f"logged in") # us                  automatically
…
  try
    di                                                c)
    pe                                                n, c)
  exce
    lo
```

```
class UsernameScope:
    def __init__(username):
        _usernameHolder.username = username
    def __enter__(self):
        pass
    def __exit__(self):
        _usernameHolder.username = None
```

LAB: Setting up the environment

3 min

1. Open a command line / shell / terminal
2. Change directory into your exercises directory
3. Create a directory within called logex: mkdir logex
4. Change directory into logex: cd logex
5. Clone the source code from Github: git clone https://github.com/SecureFromScratch/HalfDay2023/
6. Switch branches: git switch log_extras

LAB: Add Missing Logging Information

3 min

1. Execute the tasks server
2. Connect with the client and see what is the log output
3. Perform actions as different users and check the log
4. Why do we need UsernameScope to be used with a: with (python), try-with-resources (jave), using (c#)
    1. Try updating the username manually

Take Aways

Logging is Essential to Security

Must Contain all Relevant Information

An Error/Exception Must Be Logged

Part 7:
Sensitive Information in Logs

Sensitive Information in Logs

AKA Information Disclosure

AKA Information Leakage

AKA Exposing sensitive information

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

$\boxed{\textbf{T}\text{rust Boundaries}}$ → Logs are not secure

Sensitive Information - Examples

- Email
- ID
- ID Issue Date
- Birthdate
- Phone
- …

- Address
- Credit card number
- Pet name
- Family members
- First Name?
- Full Name?

- **Username?** - is there danger here?

Sensitive Information in Logs – How to Hide

- Don’t use it in log
- Hash it: whytsoft@gmail.com → 0xFD34A78BC22A5326
- Obscure it: wh****ft@gmail.com
- Replace it with an arbitrary id

LAB: Obscure the Username

7 min

1. Obscure the username in the Tasks Server log
   Example: yariv → ya***v
2. How many places in code did you need to change?
3. Check again.
   Search the logs
   Is the username really no longer in the log?

Obscure the Username

Did it succeed?

How?

Is this a good, general solution?

<<DISCUSSION>>

Secure From Scratch

Take Aways

Logging is Essential to Security

Must Contain all Relevant Information

… but Avoid Sensitive Data Leakage

An Error/Exception Must Be Logged

Part 8:
Pii Building Block

Secure From Scratch

Manual Labor

Hard

Boring

Tedious

Error Prone

I would rather write interesting code!

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

Trust Boundaries

Pii Class

Requirements?

<<DISCUSSION>>

Secure From Scratch

Pii Class

Obscured when logged

Exposed where needed

LAB: Writing a Pii Class

15 min

Write a class called Pii

1. It can be initialized with string data
2. When output to log it automatically obscures its data
3. There's a method for accessing the sensitive data
4. Integrate the Pii class into the logging extras machanism
5. But what about SimpleServer?

The Need for Concatenation

<<Code Overview>>

PiiConcat

A Lazy/Delayed Concatenation Mechanism

Holds List of Unconcatenated Values

Concatenates on Demand

Output Encoding Decided at Time of Output

Secure From Scratch

PiiConcat

<<Code Overview>>

Using PiiConcat

<<Code Overview>>

Pii/PiiConcat And The Real World ☹

- Pii classes usage is invasive
- Pii classes usage is pervasive

Pii/PiiConcat And The Real World ☹

Pii classes usage is invasive

Pii classes usage is pervasive

It’s tempting to make it otherwise…

Pii/PiiConcat And The Real World ☹

Pii classes usage is invasive

Pii classes usage is pervasive

It’s tempting to make it otherwise…

… but sometimes impossible,

Pii/PiiConcat And The Real World ☹

Pii classes usage is invasive

Pii classes usage is pervasive

It’s tempting to make it otherwise…

… but sometimes impossible,

… sometimes unavoidable,

Pii/PiiConcat And The Real World ☹

Pii classes usage is invasive

Pii classes usage is pervasive

It’s tempting to make it otherwise…

… but sometimes impossible,

… sometimes unavoidable,

and *always* dangerous.

Pii/PiiConcat And The Real World ☺☹

- Piecemeal integration
- Conversion at integration boundaries
- It’s going to take time
- It’s dangerous – but not more than existing code

Summary

Summary

Access Control/Authorization

Dealing with User Input

- Welcome/Allow Lists

Logging

- Logging as a Security Tool
- Sensitive Information in Logs

PREVENT

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exceptions

Neat code

Trust Boundaries

Call For Action

- Deepen your PREVENT
- Contribute PREVENT libraries & components
- Implement PREVENT @ work
- Share your experiences

Secure From Scratch

YouTube Channel

https://youtube.com/@SecureFromScratch

LinkedIn Page

https://www.linkedin.com/company/secure-from-scratch/

Open Source GIT

https://github.com/**SecureFromScratch/HalfDay2023**