# Họ và tên: Trần Minh Triết

# MSSV: SE172241

# Class: IA1702

# Question:

**What is an OS Command Injection vulnerability, and how does it differ from other types of injection attacks?**

OS Command Injection (OCI) is a type of security vulnerability that occurs when an application allows an attacker to execute arbitrary operating system commands on the host operating system. This type of vulnerability typically arises when an application takes user input and directly uses it to construct a command that is then passed to the underlying operating system. If the input is not properly validated or sanitized, an attacker can manipulate it to inject malicious commands.

The key characteristic of OS Command Injection is the ability for an attacker to execute commands directly on the underlying operating system. This can lead to unauthorized access, data manipulation, or other malicious activities depending on the permissions of the process executing the injected commands.

OS Command Injection differs from other types of injection attacks, such as SQL Injection or Cross-Site Scripting (XSS), in the context of the attack vector and the targeted system components. Here's a brief comparison:

* SQL Injection: SQL Injection attacks target databases by manipulating SQL queries. Attackers inject malicious SQL code into input fields, and if the application fails to properly validate or sanitize this input, it can lead to unauthorized access, data manipulation, or even deletion within the database.
* Cross-Site Scripting (XSS): XSS attacks occur when attackers inject malicious scripts into web pages that are then executed by the victim's browser. Unlike OS Command Injection, XSS does not involve executing commands on the server's operating system. Instead, it focuses on manipulating the content served to users on the client side.

In summary, OS Command Injection involves manipulating commands that are executed on the host operating system, making it a distinct type of vulnerability with potentially severe consequences if not properly mitigated.

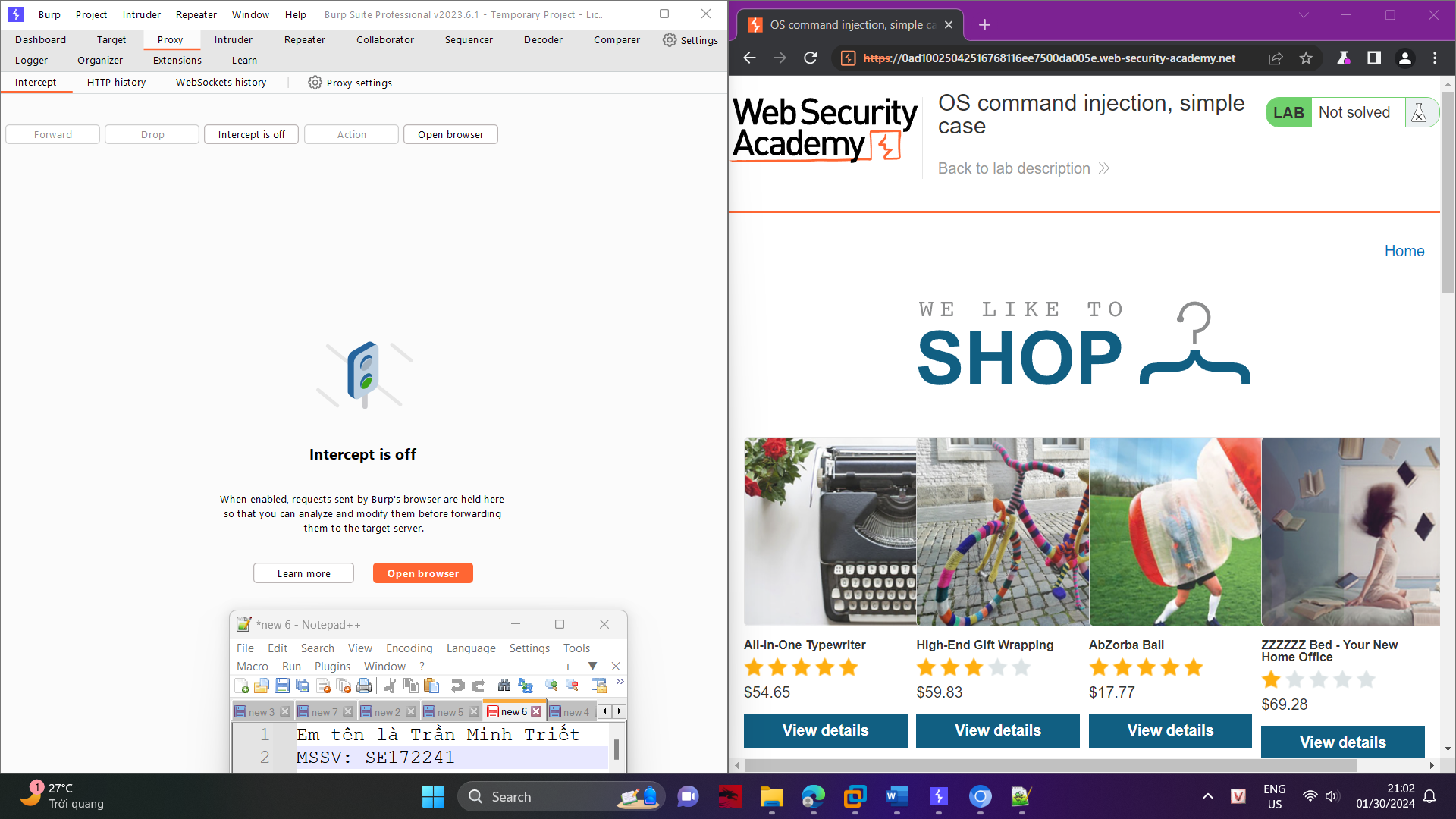
**Explain how an attacker can exploit this vulnerability to execute unauthorized commands on the server’s operating system and discuss the potential impact of such an attack on an application's security and data integrity.**

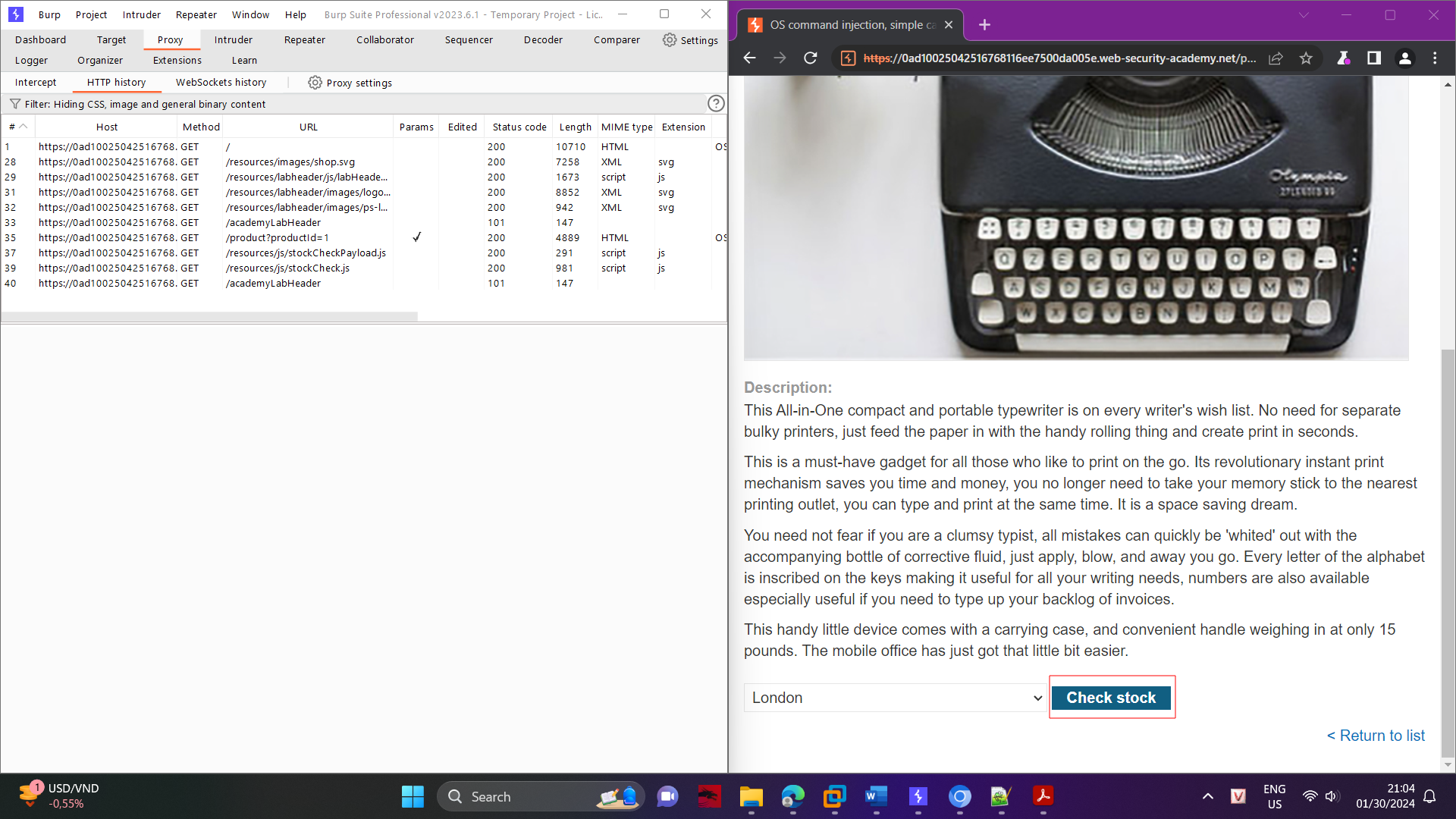
An attacker can exploit an OS Command Injection vulnerability by manipulating input parameters that are directly used in constructing operating system commands. When an application doesn't properly validate or sanitize user input before using it to build commands, an attacker can inject malicious commands, leading to unauthorized execution of arbitrary code on the server's operating system. Here's a step-by-step explanation of how an attacker might exploit this vulnerability:

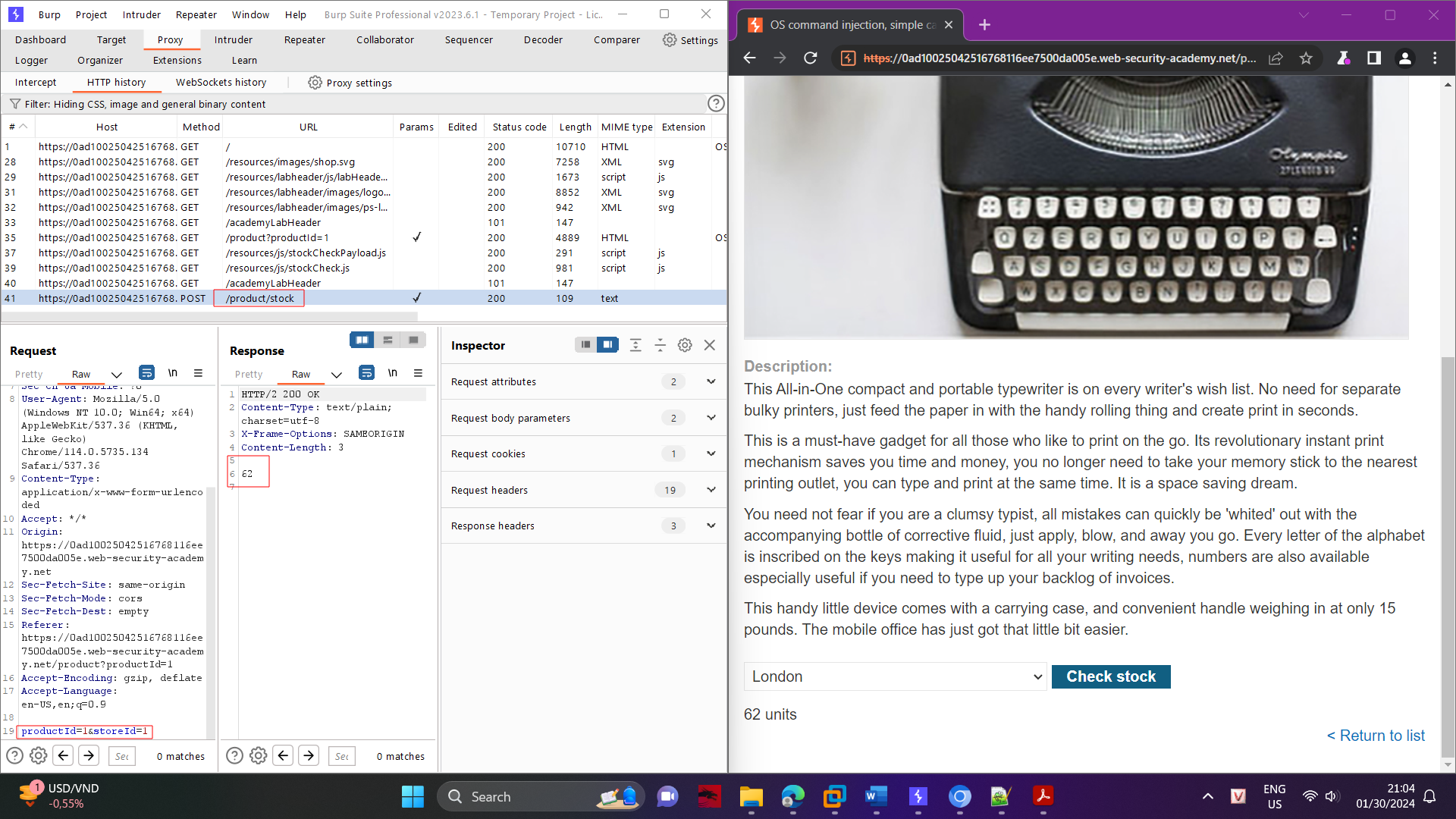
* Identifying the Vulnerability: The attacker identifies a web application or system that processes user input to construct operating system commands. This could be in the context of a search function, file upload feature, or any other functionality where user input is used to form commands.
* Crafting Malicious Input: The attacker provides input that includes special characters, such as semicolons (;), pipes (|), or backticks (`), to inject additional commands into the original one.
* Executing Unauthorized Commands: The application, without proper input validation, constructs a command by concatenating the user input with the original command. The injected command runs alongside the legitimate command, leading to the execution of unauthorized commands on the server's operating system.

**Lab 12: OS command injection, simple case**

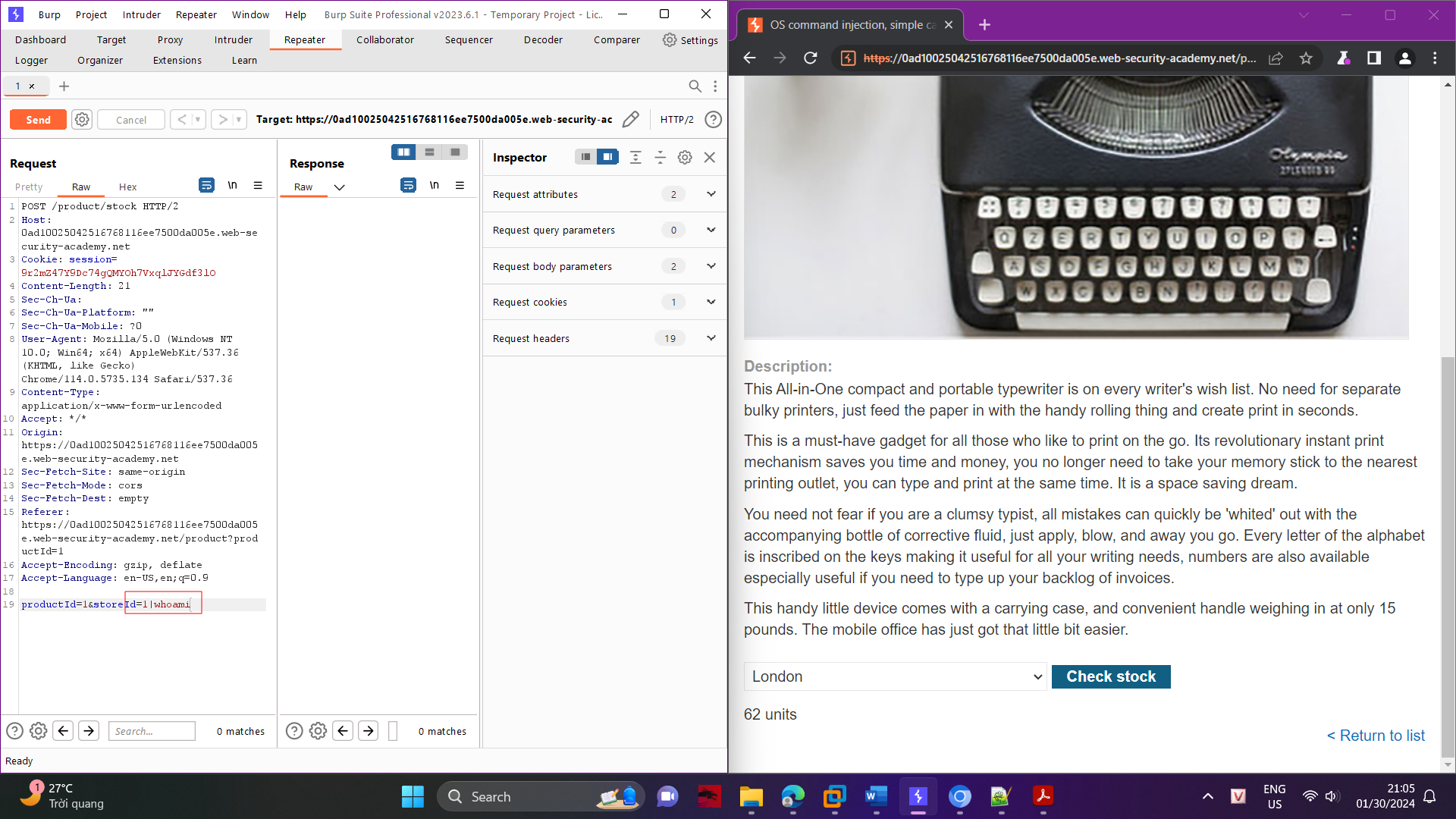
Use Burp Suite to intercept and modify a request that checks the stock level.







Modify the storeID parameter, giving it the value 1|whoami.



Observe that the response contains the name of the current user.

