5. Command Injection

Objective: To learn how command injection attack works

Tools: DVWA, Burp Suit, custom vulnerable web application

Command injection is a critical security vulnerability that occurs when an attacker can execute arbitrary commands on a host operating system via a vulnerable application. This type of attack typically targets applications that pass user-supplied data to system commands without adequate input validation or sanitation. Command injection can lead to unauthorized system access, data exfiltration, privilege escalation, and sometimes even full system compromise.

How Command Injection Works

Command injection vulnerabilities occur in applications that use user input within system-level commands without properly validating or escaping that input. Attackers can manipulate the input to execute unintended commands, leveraging characters like ;, &&, |, or & to chain commands.

Types of Command Injection

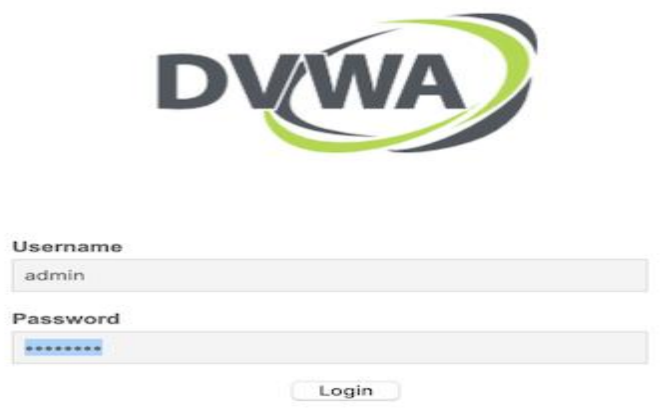
1. **Shell Injection**: Directly injects commands into shell scripts.
2. **OS Command Injection**: Targets applications that execute system commands on the operating system.
3. **Arbitrary Code Execution**: Allows attackers to run code in various languages within the application, often leading to severe consequences.

Mitigation Techniques

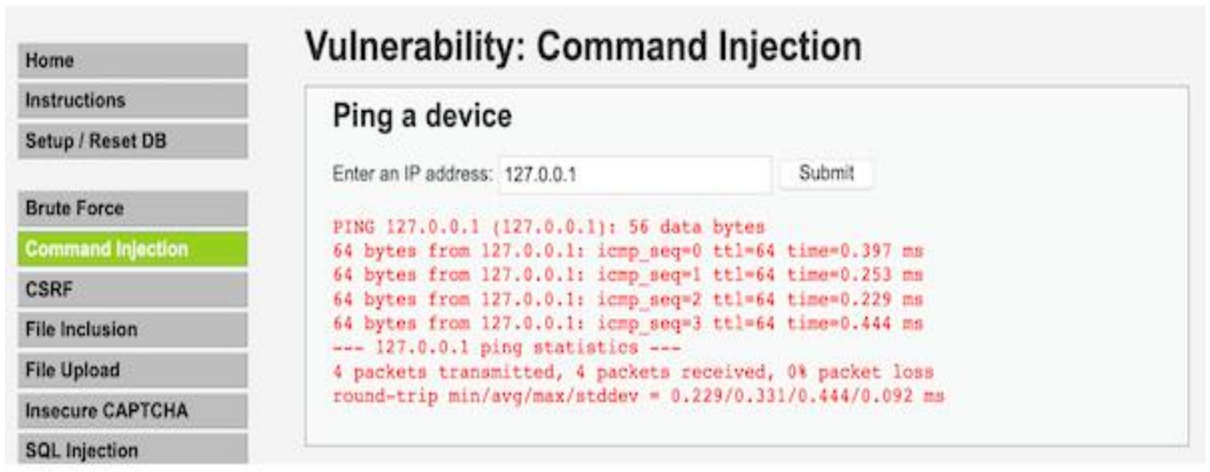
1. **Input Validation and Sanitization**: Validate and sanitize all user input, ensuring it only contains expected values.
2. **Parameterized Commands**: Use parameterized functions instead of concatenating user input directly into command strings.
3. **Use APIs Over Direct System Calls**: Instead of using system commands, opt for language-specific libraries or APIs to perform operations.
4. **Least Privilege Principle**: Limit the permissions of applications that run system commands to prevent privilege escalation.
5. **Escaping Special Characters**: Escape any potentially harmful characters in the input to prevent chaining commands.

CSE-CY, RNSIT

1. You may login with user name **admin** and password **password**.



2. Access the **Command Injection** page using the menu on the left. It will let you specify a IP adddress (e.g. **127.0.0.1**) such that the DVWA server executes the **ping** command internally to that IP, and then reports the output of the ping command.



```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.397 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.253 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.229 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.444 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.229/0.331/0.444/0.092 ms
```

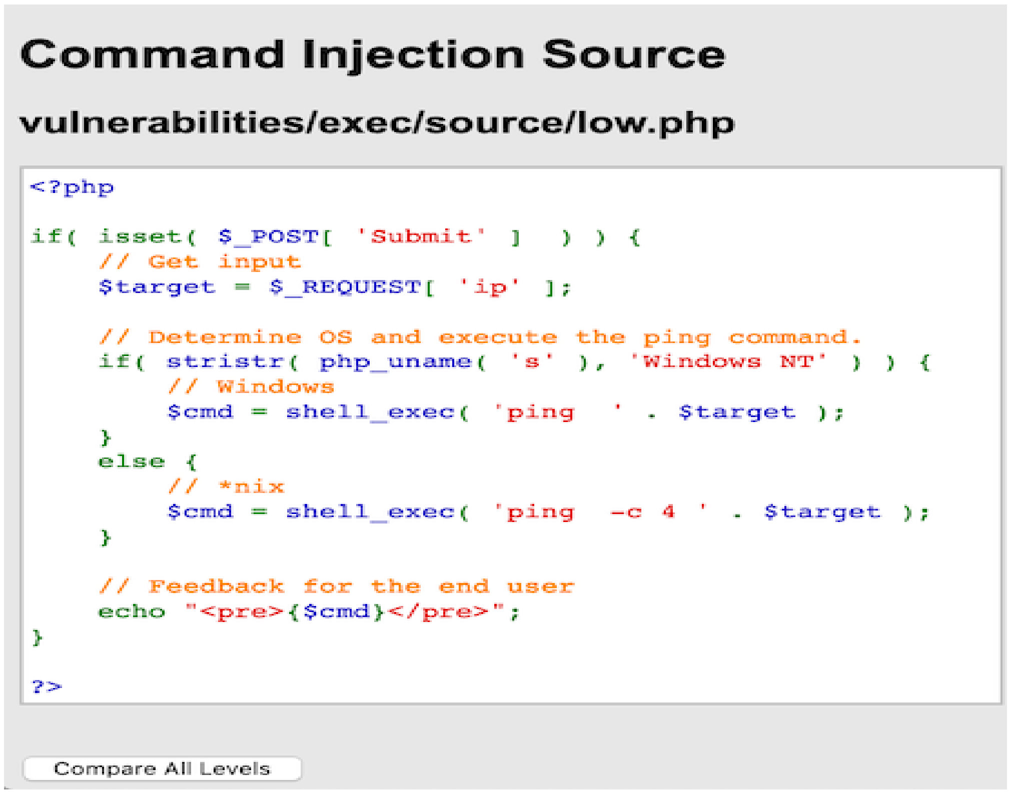
3. Exploit vulnerabilities by supplying malicious input that leads to ("injects") the execution of command

```
cat /etc/passwd
```

RNSIT

Scroll down the page and click on **View Source** to observe the code that is is executed on the server side. Analyze the code to understand the input validation mechanism (if any). You can also click on **View Help** for an explanation of the input validation mechanisms and vulnerability exploitation hints.

- Source code window:

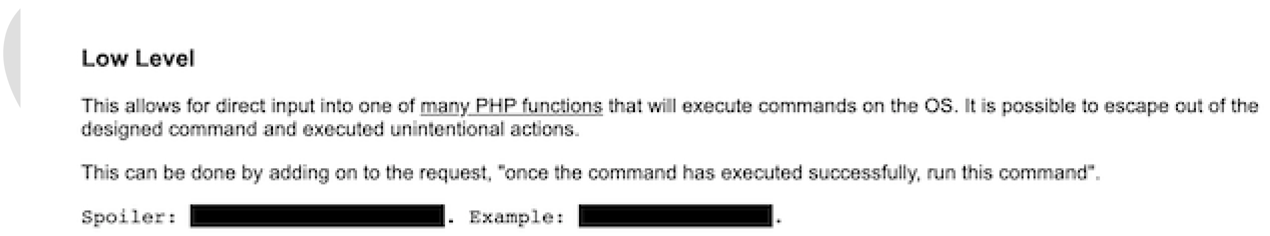


Command Injection Source

vulnerabilities/exec/source/low.php

```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

- Help page fragment:



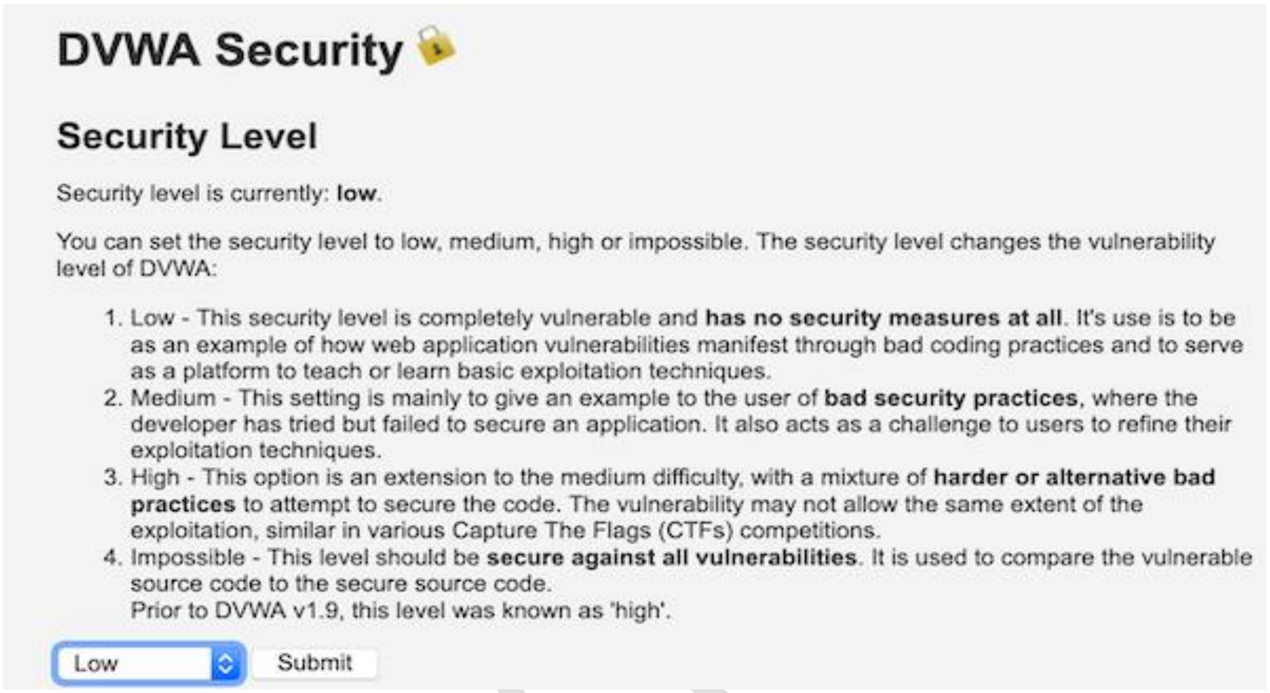
Low Level

This allows for direct input into one of many PHP functions that will execute commands on the OS. It is possible to escape out of the designed command and executed unintentional actions.

This can be done by adding on to the request, "once the command has executed successfully, run this command".

Spoiler: [redacted]. Example: [redacted].

4. Change the DVWA Security Level, initially set to **Low**, and **repeat step 4** for the **Medium**, **High** and finally the **Impossible** security levels.



DVWA Security

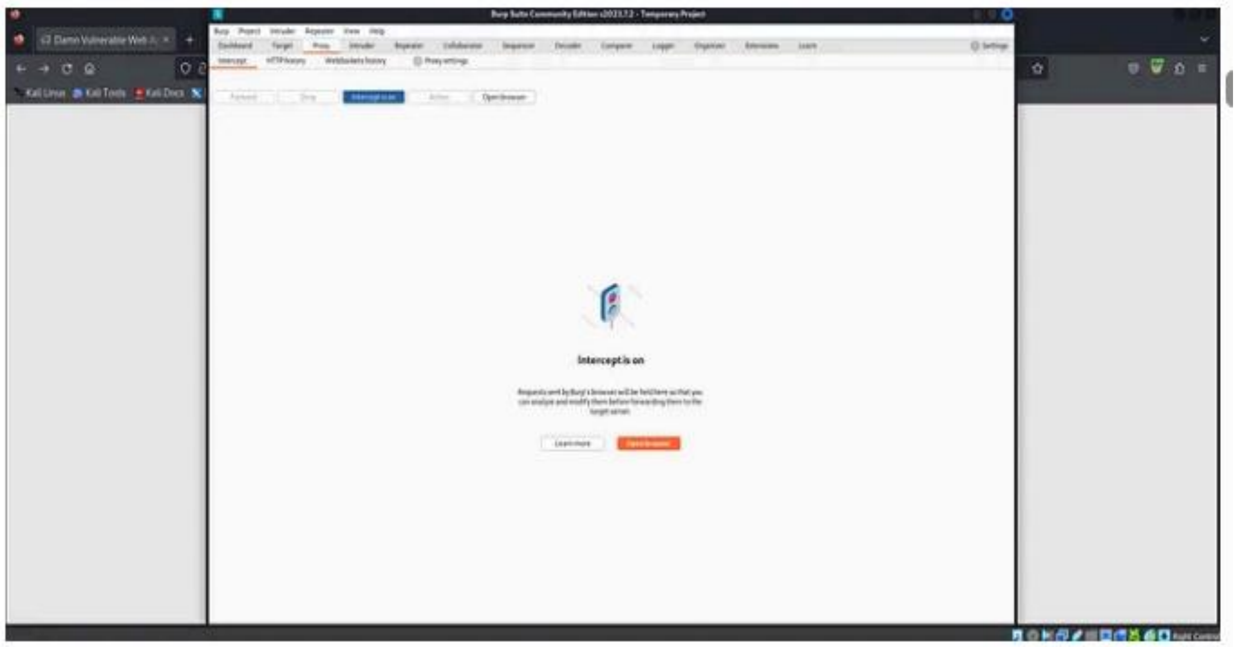
Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Prior to DVWA v1.9, this level was known as 'high'.

Low | Submit



Requests sent by Burp's browser will be held here so that you can analyze and modify them before forwarding them to the target server.

Link: https://www.dcc.fc.up.pt/~edrdo/aulas/qses/lectures/lab1/