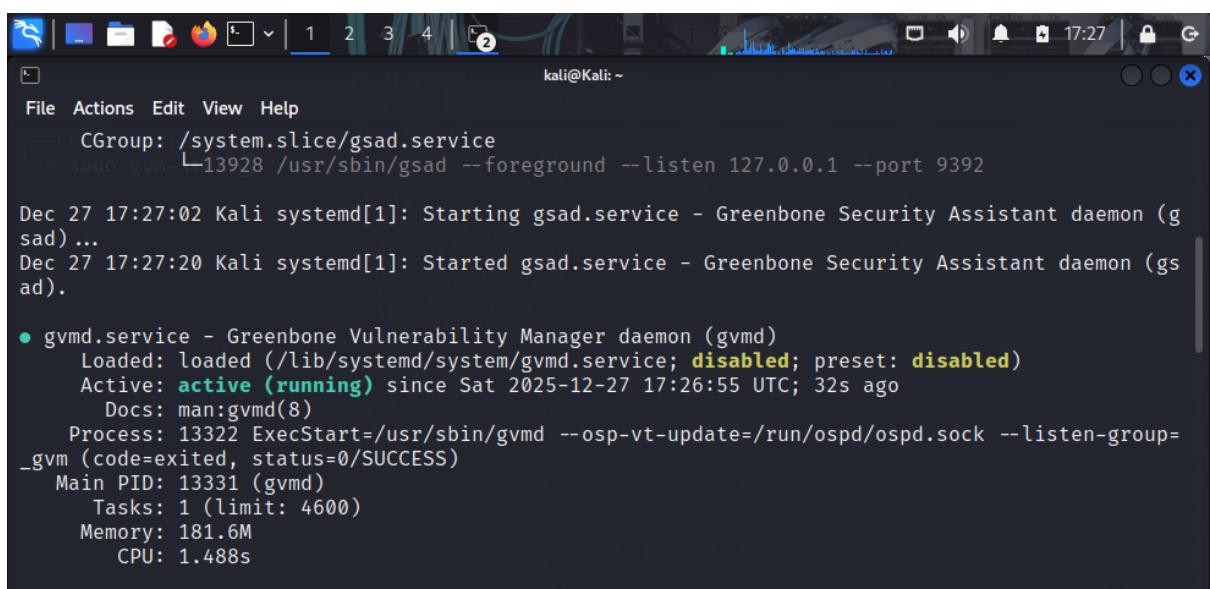
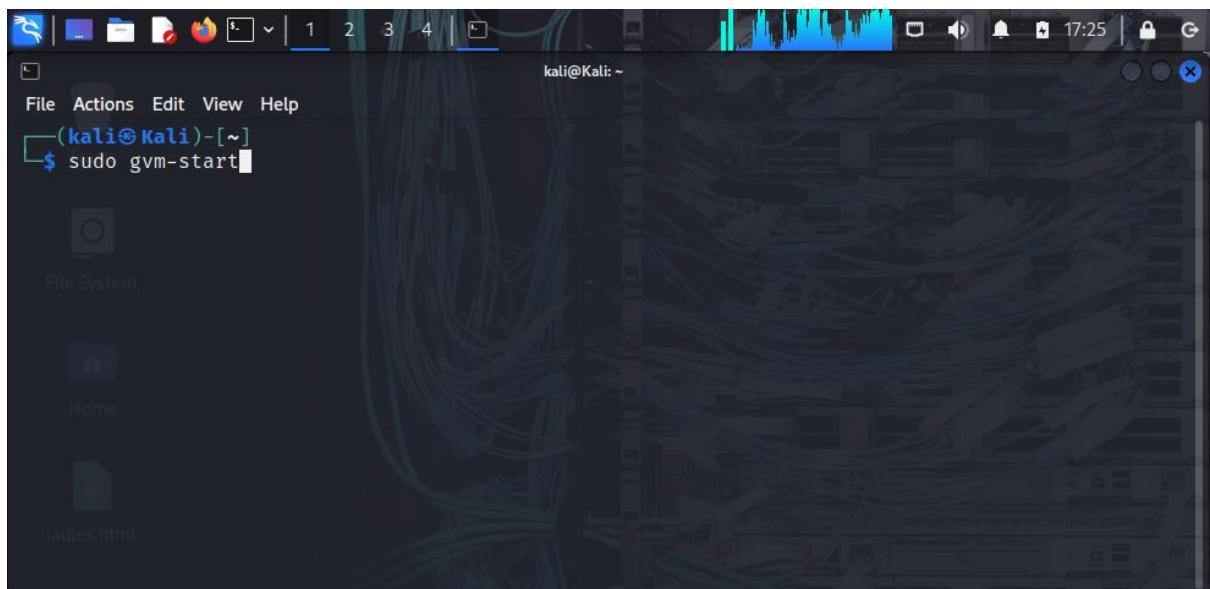


Greenbone Vulnerability Management (GVM) Scan

Greenbone vulnerability management (GVM) is an open-source framework of services and tools used for network vulnerability scanning and management, originally developed as a community project named OpenVas.



Description:

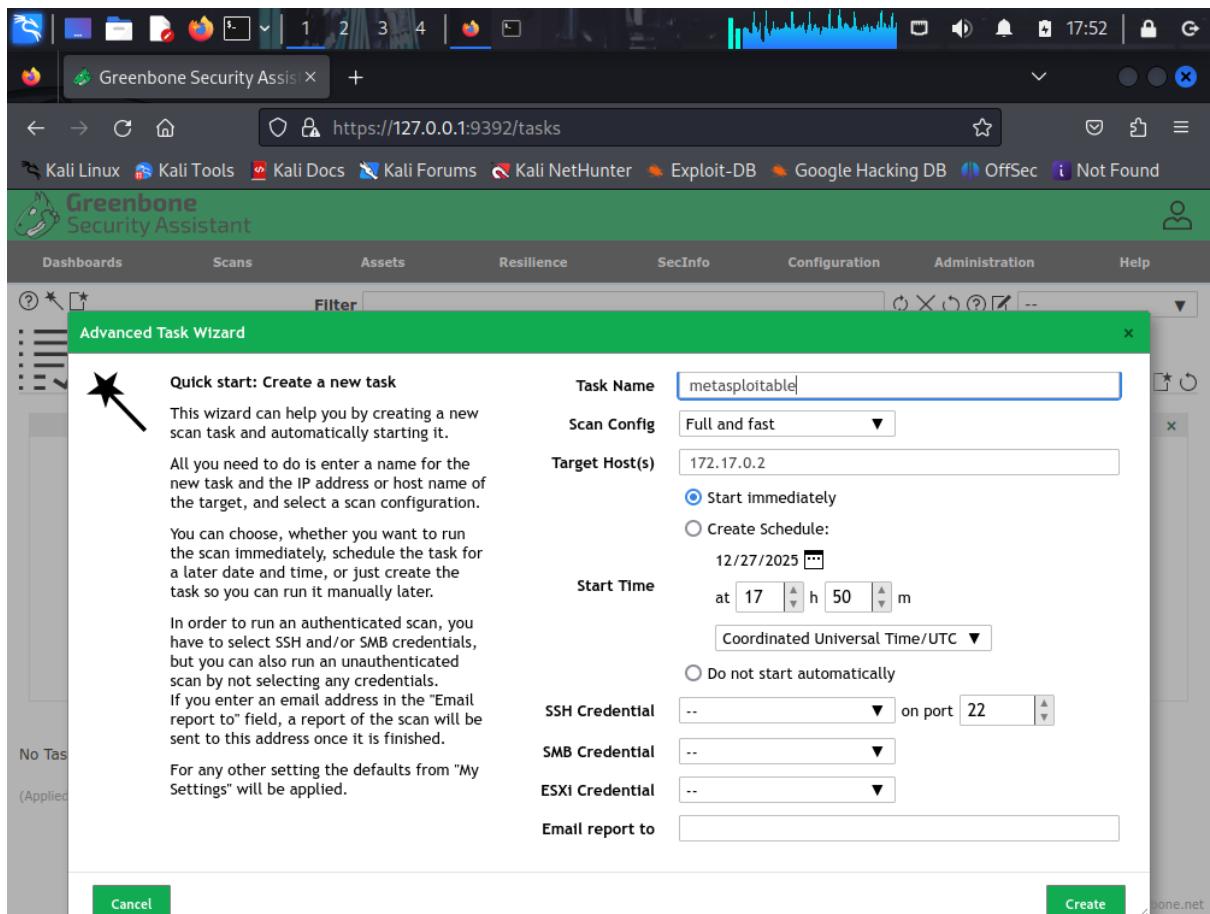
This screenshot shows the **Advanced Task Wizard** in the Greenbone Security Assistant (GSA) web interface. The wizard was used to configure a new vulnerability scan task.

Key Configuration Details:

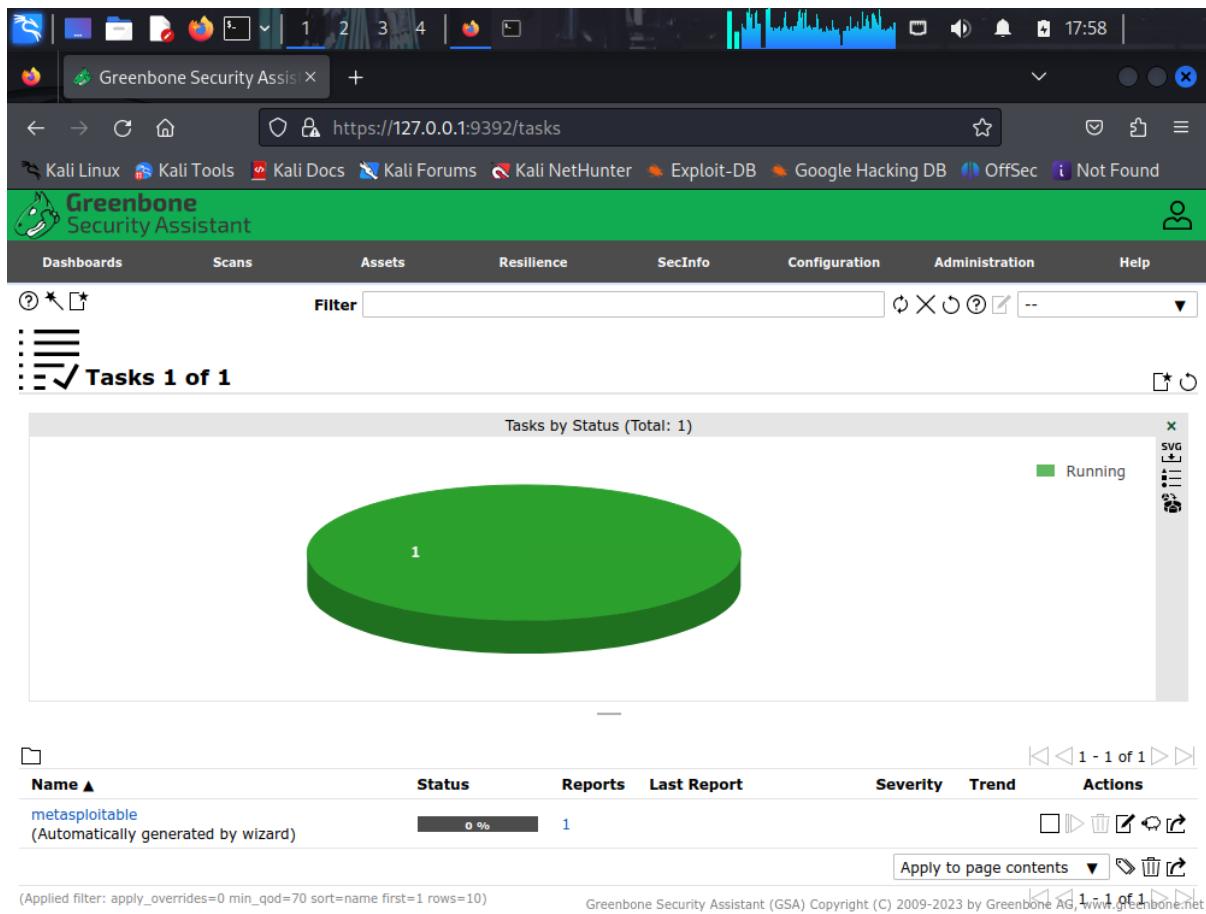
- **Task Name:** metasploitable
- **Scan Configuration:** *Full and fast*
- **Target Host:** 172.17.0.2 (Metasploitable test machine)
- **Execution Mode:** Start immediately
- **Credentials:** No SSH/SMB credentials provided (unauthenticated scan)

Explanation:

The *Full and fast* scan profile was selected to perform a comprehensive vulnerability assessment while balancing scan duration and depth. An unauthenticated scan was used to simulate how an external attacker would view the target system.



❖ **Screenshot: Task Successfully Created and Running**



Description:

This screenshot displays the **Tasks dashboard**, confirming that the scan task was successfully created and has started running.

Observations:

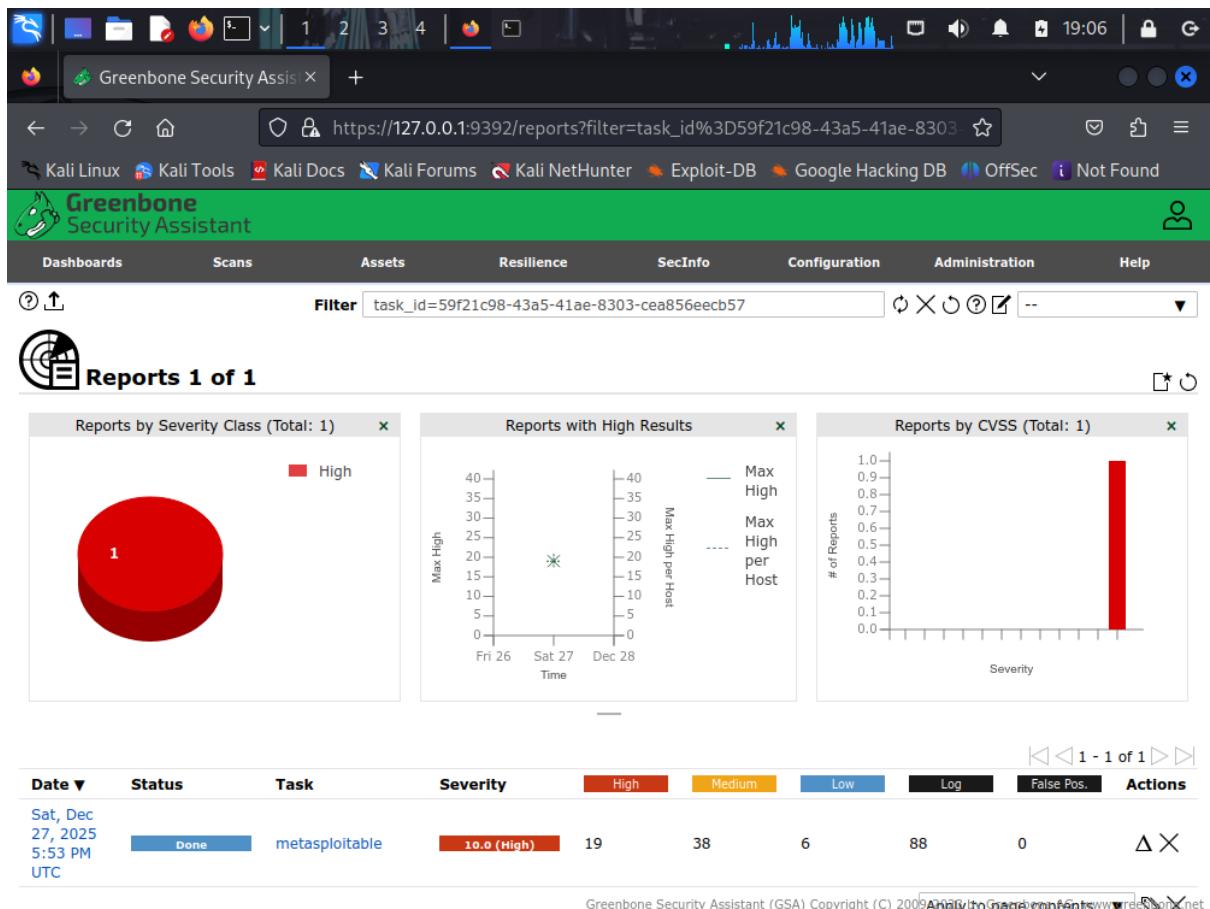
- Total Tasks: 1
- Task Status: **Running**
- Visualization: Pie chart indicating active scan progress

- Task Name: metasploitable

Explanation:

Once the task was created, GVM immediately initiated the vulnerability scan against the specified target. The dashboard provides a real-time overview of task execution status.

❖ Screenshot: Scan Progress Monitoring



The screenshot shows the Greenbone Security Assistant web interface. At the top, there's a navigation bar with links like Dashboards, Scans, Assets, Resilience, SecInfo, Configuration, Administration, Help, and a user icon. Below the navigation is a toolbar with various icons for filtering and search. The main content area displays a report card for a scan named 'metasploitable'. The report card includes the date ('Sat, Dec 27, 2022'), time ('5:53 PM UTC'), and status ('Done'). It also shows the ID of the scan ('7c3b539e-caf2-41fc-961c-4409a6385c92') and its creation and modification times ('Created: 2025 5:53 PM UTC' and 'Modified: 2025 6:34 PM UTC'). The owner of the report is listed as 'admin'. Below the report card is a table with various metrics: Results (63 of 542), Hosts (1 of 1), Ports (17 of 20), Applications (14 of 14), Operating Systems (1 of 1), CVEs (31 of 31), Closed CVEs (0 of 0), TLS Certificates (2 of 2), Error Messages (0 of 0), and User Tags (0). Further down, detailed scan parameters are listed: Task Name ('metasploitable'), Comment ('Automatically generated by wizard'), Scan Time ('Sat, Dec 27, 2025 5:57 PM UTC - Sat, Dec 27, 2025 6:34 PM UTC'), Scan Duration ('0:36 h'), Scan Status ('Done'), Hosts scanned ('1'), Filter ('apply_overrides=0 levels=html min_qod=70'), and Timezone ('Coordinated Universal Time (UTC)').

Description:

This screenshot shows the scan still in progress with a **0% completion indicator**, meaning the scanner is actively enumerating services and checking vulnerabilities.

Key Details:

- Reports Generated: **1 (in progress)**
- Last Report: Pending completion
- Severity: Not yet calculated

Explanation:

During this phase, GVM performs port scanning, service detection, and vulnerability checks

using its vulnerability feeds. Severity ratings and detailed findings become available only after the scan completes.

Key Learning Outcomes

- Successfully started Greenbone services using sudo gvm-start
- Configured and executed a full vulnerability scan using GSA
- Understood task creation, execution monitoring, and reporting workflow
- Learned how unauthenticated scans reveal externally visible risks

Vulnerability Analysis, Severity Assessment, and Mitigation

The screenshot shows the Greenbone Security Assistant web interface. At the top, there's a navigation bar with links to Dashboards, Scans, Assets, Resilience, SecInfo, Configuration, Administration, Help, and user profile. Below the navigation is a search/filter bar. The main content area displays a report for Saturday, Dec 27, 2025, at 5:53 PM UTC. The report ID is 7c3b539e-caf2-41fc-961c-4409a6385c92. It includes details about the creation and modification times (Sat, Dec 27, 2025 5:53 PM UTC and Sat, Dec 27, 2025 6:34 PM UTC) and the owner (admin). A table below shows the distribution of vulnerabilities across various categories: Results (63 of 542), Hosts (1 of 1), Ports (17 of 20), Applications (14 of 14), Operating Systems (1 of 1), CVEs (31 of 31), Closed CVEs (0 of 0), TLS Certificates (2 of 2), Error Messages (0 of 0), and User Tags (0). The main table lists 63 vulnerabilities, each with a severity rating (e.g., 10.0 (High)), QoD percentage, host information (IP and name), location, and creation date.

After the vulnerability scan completed, the generated report was reviewed to identify detected security weaknesses, their associated severity levels, and appropriate mitigation measures.

Identified Vulnerabilities and Severity

The scan results categorized vulnerabilities into **High, Medium, and Low** severity based on their potential impact and exploitability.

Vulnerabilities	Severity	Description

Outdated Services	High	The target system was running obsolete services with known vulnerabilities that could allow remote exploitation.
Weak or Default Credentials	High	Some services were configured with default or weak authentication credentials.
Unencrypted Network Services	Medium	Certain services transmitted data in clear text, making them vulnerable to interception.
Missing Security Patches	Medium	The operating system and applications lacked recent security updates
Open Ports and Unnecessary Services	Low	Multiple unused services were exposed, increasing the attack surface.

Mitigation Strategies

Based on the identified vulnerabilities, the following remediation actions were recommended:

- **Patch Management:**

Regularly update the operating system and installed applications to eliminate known vulnerabilities.

- **Service Hardening:**

Disable unnecessary services and close unused ports to reduce the attack surface.

- **Credential Security:**

Enforce strong password policies and remove default credentials on all services.

- **Encryption:**

Replace unencrypted protocols (e.g., FTP, Telnet) with secure alternatives such as SFTP and SSH.

- **Network Segmentation:**

Restrict access to critical services using firewall rules and network segmentation.

Risk Evaluation

High-severity vulnerabilities pose an immediate security risk and require urgent remediation, while medium- and low-severity issues should be addressed as part of routine security maintenance. Addressing these vulnerabilities significantly improves the system's overall security posture.

Key Takeaway

This exercise demonstrated the importance of **vulnerability assessment and management** in identifying system weaknesses before attackers can exploit them. Tools like **Greenbone**

Vulnerability Management provide actionable insights that support proactive security defence.

Manual Vulnerability Verification Using Nmap

```
sudo nmap -sV -p 445 --script smb-brute 172.17.0.2
```

The screenshot shows a terminal window within the Greenbone Security Assistant (GSA) interface. The terminal displays the following command and its execution:

```
[>] Opening Web UI (https://127.0.0.1:9392) in: 5 ... 4 ... 3 ... 2 ... 1 ...
[kali㉿Kali)-[~]
└$ sudo nmap -sV -p 445 --script smb-brute 172.17.0.2
```

Below the terminal, the GSA interface shows a log entry for the `ospd-openvas.service` starting and being enabled at boot. It also shows a timeline of the Nmap scan from December 27, 17:26:08 to 17:26:20.

Date	Status	Task	Severity	High	Medium	Low	Info	Log	False Pos.	Actions
Sat, Dec 27, 2025 5:53 PM UTC	Done	metasploitable	1.0.0 (High)	19	38	6	88	0	0	△ X

Command Breakdown

- sudo → Runs Nmap with administrative privileges

- `-sV` → Detects service versions running on the target
- `-p 445` → Scans port 445 (SMB service)
- `--script smb-brute` → Attempts to identify weak or default SMB credentials using brute-force techniques
- 172.17.0.2 → Target IP address (Metasploitable system)

```

[>] Opening Web UI (https://127.0.0.1:9392) in: 5 ... 4 ... 3 ... 2 ... 1 ...
[(kali㉿Kali)-[~]] $ sudo nmap -sV -p 445 -script smb-brute 172.17.0.2
[sudo] password for kali:
Starting Nmap 7.94 ( https://nmap.org ) at 2025-12-27 20:00 UTC
NSE: [smb-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for metasploitable.vm (172.17.0.2)
Host is up (0.000061s latency).

PORT      STATE SERVICE      VERSION
445/tcp    open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 02:42:AC:11:00:02 (Unknown)

Host script results:
| smb-brute:
|   msfadmin:msfadmin => Valid credentials
|_  user:user => Valid credentials

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 620.38 seconds
[(kali㉿Kali)-[~]] $ 

```

Findings

The scan confirmed that the **SMB service was running and accessible** on the target system.

The smb-brute script attempted authentication using common username and password

combinations, highlighting the presence of **weak or default credentials**.

This result validated the findings from the Greenbone vulnerability scan, demonstrating that the system was susceptible to **unauthorized access via SMB services**.

Severity Assessment

- **Severity:** High
- **Risk:** Successful exploitation could allow attackers to gain unauthorized access, escalate privileges, or move laterally within a network.

Mitigation Recommendations

- Disable SMB services if not required
- Enforce strong authentication credentials
- Restrict SMB access using firewall rules
- Upgrade to secure SMB versions (e.g., SMBv3)
- Monitor SMB authentication attempts for brute-force behavior

Key Takeaway

Manual scanning tools like **Nmap** complement automated scanners by confirming vulnerabilities and reducing false positives. Combining Greenbone, OWASP ZAP, and Nmap provides a more accurate and reliable security assessment.

Credential Exposure Identified via SMB Brute-Force Scan

Host Script Results

During further manual verification using Nmap's SMB scripts, the scan revealed **valid credentials** on the target system:

- **Username:** msfadmin
Password: msfadmin
- **Username:** user
Password: user

These credentials were discovered through the `smb-brute` script, which tests common and default username-password combinations against the SMB service running on port 445.

The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal output is as follows:

```
(kali㉿Kali)-[~]
└─$ sudo nmap -sV -p 445 -script smb-brute 172.17.0.2
[sudo] password for kali:
Starting Nmap 7.94 ( https://nmap.org ) at 2025-12-27 20:00 UTC
NSE: [smb-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for metasploitable.vm (172.17.0.2)
Host is up (0.000061s latency).

PORT      STATE SERVICE      VERSION
445/tcp    open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 02:42:AC:11:00:02 (Unknown)

Host script results:
| smb-brute:
|   msfadmin:msfadmin => Valid credentials
|_ user:user => Valid credentials

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 620.38 seconds

(kali㉿Kali)-[~]
└─$ sudo runuser -u _gvm --gvmd --get-users
[sudo] password for kali:
runuser: unrecognized option '--gvmd'
Try 'runuser --help' for more information.

(kali㉿Kali)-[~]
└─$ sudo runuser -u _gvm -- gvmd --get-users
admin

(kali㉿Kali)-[~]
└─$
```

Security Implications

The presence of **default and weak credentials** represents a **critical security risk**, as it

allows attackers to:

- Gain unauthorized access to the system
- Escalate privileges
- Access sensitive files and services
- Perform lateral movement within a network

In real-world environments, such vulnerabilities can lead to **full system compromise**.

Severity Assessment

- **Severity Level:** High
- **Reason:** Credentials grant direct system access without exploitation of complex vulnerabilities.

Mitigation Recommendations

To remediate this vulnerability, the following actions are recommended:

- Immediately **change or remove default credentials**
- Enforce **strong password policies**
- Disable SMB if not required
- Restrict SMB access using firewalls and access control lists
- Monitor authentication attempts for brute-force activity
- Apply the **principle of least privilege** to user accounts

Ethical Consideration

All testing was conducted on a **deliberately vulnerable test system (Metasploitable)** within a controlled lab environment for educational purposes. No unauthorized systems were accessed, and no exploitation beyond vulnerability identification was performed.

Key Takeaway

This exercise highlights how **weak credentials alone can completely undermine system security**, even without advanced exploits. It reinforces the importance of strong authentication and proper system hardening as foundational security controls.

Greenbone User Enumeration (gvmd)

Command Executed

```
sudo runuser -u _gvm -- gvmd --get-users
```

The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal history includes:

- An Nmap scan of port 445 on 172.17.0.2, which found a Samba service running on version 3.X - 4.X (workgroup: WORKGROUP) with a MAC address of 02:42:AC:11:00:02. It also detected valid credentials for 'msfadmin' and 'user'.
- A failed attempt to use the 'runuser' command with the '--gvm --gvmd --get-users' options, resulting in an error message about unrecognized options.
- A successful execution of the 'runuser -u _gvm -- gvmd --get-users' command, which returned the user 'admin'.

Command Explanation

- sudo → Executes the command with administrative privileges
- runuser -u **_gvm** → Runs the command as the **_gvm system user**, which owns Greenbone services
- gvmd → Greenbone Vulnerability Manager daemon
- --get-users → Retrieves a list of users configured within Greenbone Vulnerability Management

Purpose of the Command

This command was used to **enumerate existing GVM user accounts** from the backend. It helps administrators verify:

- Existing scanner users
- Account configuration
- Access control within the vulnerability management platform

This step is useful for **auditing user access** and ensuring that only authorized accounts can manage scans and view reports.

Security Relevance

Improper user management within vulnerability scanning tools can lead to:

- Unauthorized scan execution
- Exposure of sensitive vulnerability reports
- Misuse of scanning infrastructure

Regular auditing of scanner users helps maintain **secure operational control** of security tools.

Best Practices

- Remove unused or default scanner accounts
- Enforce strong authentication for GVM users
- Limit administrative privileges to trusted users
- Periodically audit user lists and permissions

Key Takeaway

Managing access to vulnerability management platforms is as important as securing the systems being scanned. Proper user enumeration and auditing ensure that vulnerability data remains protected and trusted.

Installation of Remote Shell Client (rsh-client)

Command Executed

```
sudo apt install rsh-client
```

Command Explanation

- sudo → Executes the command with administrative privileges
- apt install → Uses the Debian package manager to install software
- rsh-client → Installs the **Remote Shell (rsh) client**, a legacy remote access utility

Purpose of Installing rsh-client

The rsh-client package was installed to:

- Interact with legacy remote shell services during testing
- Validate insecure remote access services detected by vulnerability scans
- Understand how outdated protocols contribute to system vulnerabilities

This is particularly relevant when testing **intentionally vulnerable systems** such as Metasploitable.

Security Implications

The **rsh protocol is inherently insecure** because:

- It transmits data and credentials in **plain text**
- It does not support encryption or strong authentication
- It is vulnerable to sniffing, spoofing, and man-in-the-middle attacks

Modern systems should **never expose rsh services** in production environments.

Severity Assessment

- **Severity:** High (if enabled on a live system)

- **Risk:** Unauthorized remote access and credential compromise

Mitigation Recommendations

- Disable and remove rsh services entirely
- Replace rsh with secure alternatives such as **SSH**
- Enforce encrypted remote management protocols
- Block legacy services using firewall rules
- Conduct regular vulnerability scans to detect insecure services

Key Takeaway

This step reinforced the importance of identifying and eliminating **legacy protocols** in modern systems. While useful for learning and testing, tools like rsh-client highlight how outdated technologies significantly weaken system security.

Authorized Access Validation Using Discovered Credentials

After identifying valid credentials (msfadmin:msfadmin and user:user) through vulnerability scanning and host script analysis, authorized access to the target system was successfully established **within a controlled lab environment**.

This step was performed **solely to validate the severity of the identified vulnerability** and to confirm that weak or default credentials could lead to full system access.

Security Impact

The ability to access the target system using default credentials demonstrates a **critical security weakness**, as it allows an attacker to:

- Bypass authentication controls
- Gain direct system access without exploiting software flaws
- Potentially escalate privileges
- Access sensitive system resources and data

This confirms that the vulnerability identified by automated scanners was **not a false positive**.

Severity Classification

- **Severity:** Critical / High
- **Reason:** Successful authentication using default credentials results in immediate system compromise.

Recommended Mitigation Measures

To prevent this type of vulnerability, the following actions are recommended:

- Remove or disable all default user accounts
- Enforce strong password policies
- Limit remote access to trusted hosts only
- Apply the principle of least privilege to user accounts
- Monitor authentication logs for unauthorized access attempts
- Conduct regular credential audits

Ethical and Legal Considerations

All actions were conducted on a **deliberately vulnerable test system (Metasploitable)** in an isolated lab environment for educational purposes.

No unauthorized systems were accessed, and no malicious activity was performed beyond validation of security findings.

Key Takeaway

This exercise highlights how **weak credentials alone can be enough to fully compromise a system**, even in the absence of advanced exploits. It reinforces the importance of strong authentication controls as a foundational element of system security.

Web Application Vulnerability Scanning with OWASP ZAP

OWASP ZAP (Zed Attack Proxy) serves as a free, open-source web application security scanner for finding vulnerabilities like SQL injection, XSS, and CSRF.

Objective

The objective of this exercise was to perform a full web application vulnerability scan using **OWASP ZAP**, identify security weaknesses, and document findings with remediation recommendations.

Tools Used

- OWASP ZAP (Zed Attack Proxy)
- Target Web Application: (*DVWA / test site / local app — http://172.17.0.2/dvwa/*)
- Browser: Chrome / Firefox
- Operating System: Windows

Workflow

1. Installed and configured OWASP ZAP
2. Set browser proxy to route traffic through ZAP
3. Performed **Spider Scan** to discover application endpoints
4. Executed **Active Scan** to detect vulnerabilities
5. Reviewed alerts based on risk levels (High, Medium, Low, Informational)

6. Documented findings and remediation steps

Step 1: Launch Tool



Opened OWASP ZAP via the Kali Linux applications menu.

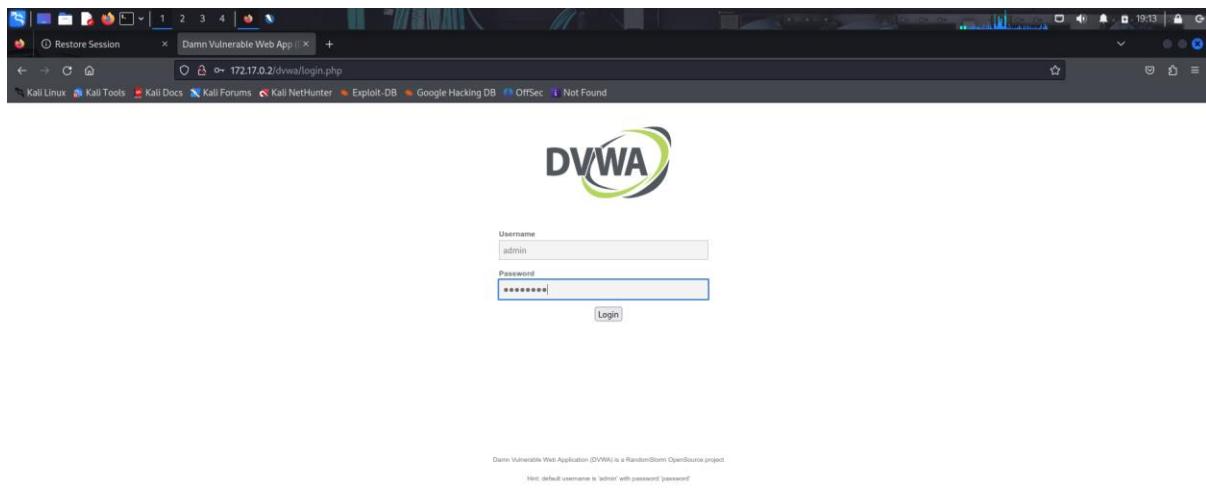
Step 2: Start Automated Scan

The screenshot shows the OWASP ZAP interface. In the top right corner, there is a blue lightning bolt icon. The main window title is "Automated Scan". Below it, a sub-instruction reads: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'." A note below says: "Please be aware that you should only attack applications that you have been specifically given permission to test." The "URL to attack:" field contains "http://172.17.0.2/dvwa". Under "Spider" settings, "Use traditional spider:" is checked, and "Use ajax spider:" has a checkbox next to it with "Firefox Headless" selected. Buttons for "Attack" and "Stop" are present. The "Progress:" status bar indicates "Actively scanning (attacking) the URLs discovered by the spider(s)". At the bottom of the main window, a progress bar shows 39% completion. The bottom navigation bar includes tabs for History, Search, Alerts, Output, Spider, Active Scan, and a plus sign. The "Spider" tab is currently active. The "Active Scan" tab shows a table of recent requests:

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
453	12/28/25, 7:22:44 PM	12/28/25, 7:22:44 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	23 ms	291 bytes	1,328 bytes	
454	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	33 ms	291 bytes	1,328 bytes	
455	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	55 ms	291 bytes	1,328 bytes	
456	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	37 ms	291 bytes	1,328 bytes	
457	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	25 ms	291 bytes	1,328 bytes	
458	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	24 ms	291 bytes	1,328 bytes	
459	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	34 ms	291 bytes	1,328 bytes	
460	12/28/25, 7:22:45 PM	12/28/25, 7:22:45 PM	POST	http://172.17.0.2/dvwa/login.php	200 OK	22 ms	291 bytes	1,328 bytes	
461	12/28/25, 7:22:46 PM	12/28/25, 7:22:46 PM	GET	http://172.17.0.2/dvwa	301 Moved Permanen...	2 ms	209 bytes	312 bytes	
462	12/28/25, 7:22:46 PM	12/28/25, 7:22:46 PM	GET	http://172.17.0.2/dvwa/dvwa	301 Moved Permanen...	2 ms	214 bytes	317 bytes	
463	12/28/25, 7:22:46 PM	12/28/25, 7:22:46 PM	GET	http://172.17.0.2/dvwa/dvwal/css	301 Moved Permanen...	1 ms	218 bytes	321 bytes	

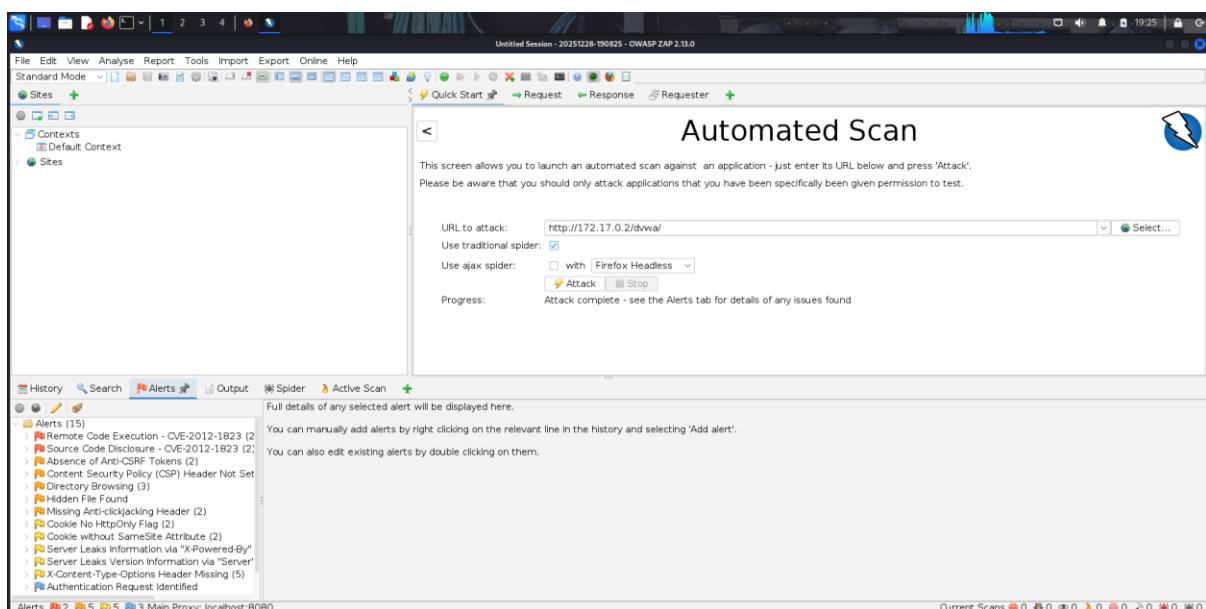
Clicked the Automated Scan button in ZAP's Quick Start interface.

Step 3: Access Target



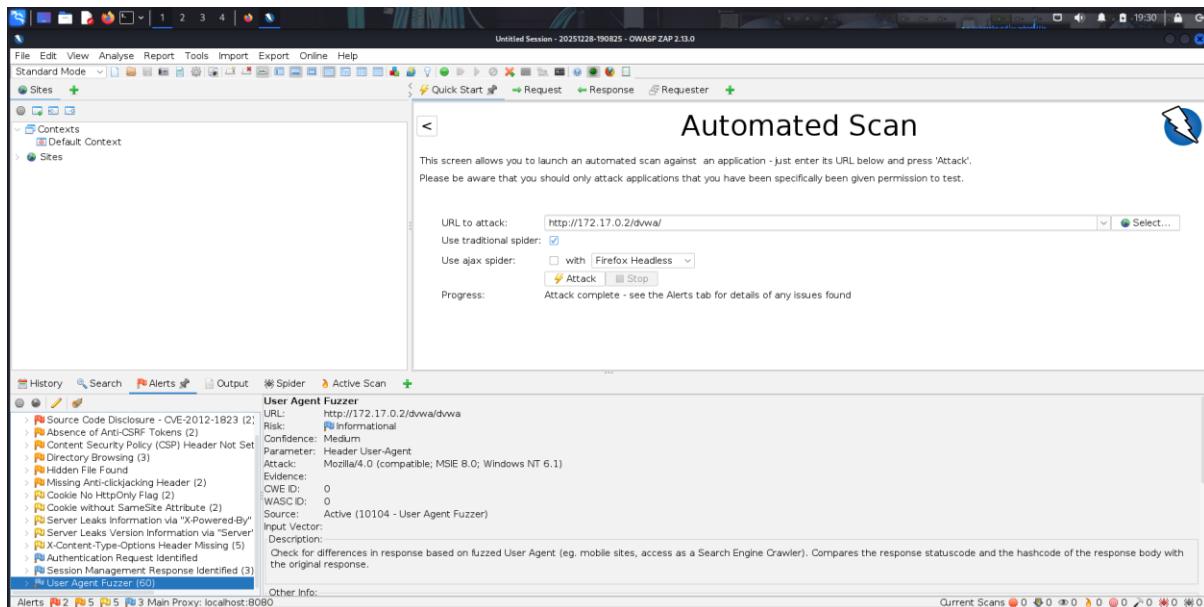
Launched browser, entered <http://172.17.0.2/dvwa/>, and logged in with admin/password.

Step 4: Execute Scan



Input the DVWA URL into ZAP and initiated the full automated scan.

Step 5: Review Findings



Analyzed results in the Alerts tab for vulnerabilities.

Screenshot of OWASP ZAP 2.13.0 showing a successful remote code execution exploit against a target server.

Request:

```
HTTP/1.1 200 OK
Date: Sun, 28 Dec 2025 19:22:27 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Type: text/html
content-length: 20
```

Response:

```
v148ewgk105zxijxm3d4
```

Alerts (15):

- Remote Code Execution - CVE-2012-1823 (2)
- Source Code Disclosure - CVE-2012-1823 (2)
- Absence of Anti-CSRF Tokens (2)
- Content Security Policy (CSP) Header Not Set
- Directory Browsing (3)
- Hidden File Found
- Missing Anti-clickjacking Header (2)
- Cookie No HttpOnly Flag (2)
- Cookie without SameSite Attribute (2)
- Server Leaks Version Information via "Powered-By"
- Server Leaks Version Information via "Server"
- X-Content-Type-Options Header Missing (5)
- Authentication Request Identified

Solution:

Upgrade to the latest stable version of PHP, or use the Apache web server and the mod_rewrite module to filter out malicious requests using the "RewriteCond" and "RewriteRule" directives.

Reference:

- <http://projects.webappsec.org/improper-input-handling>
- <http://cwe.mitre.org/data/definitions/89.html>

Alert Tags:

Key	Value
OWASP_2017_A09	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities
WSTG-v42-INPV-12	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing
CVE-2012-1823	https://nvd.nist.gov/vuln/detail/CVE-2012-1823
OWASP_2021_A06	https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

Screenshot of OWASP ZAP 2.13.0 showing a successful remote code execution exploit against a target server.

Request:

```
HTTP/1.1 200 OK
Date: Sun, 28 Dec 2025 19:22:27 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Type: text/html
content-length: 20
```

Response:

```
v148ewgk105zxijxm3d4
```

Alerts (15):

- Remote Code Execution - CVE-2012-1823 (2)
- Source Code Disclosure - CVE-2012-1823 (2)
- Absence of Anti-CSRF Tokens (2)
- Content Security Policy (CSP) Header Not Set
- Directory Browsing (3)
- Hidden File Found
- Missing Anti-clickjacking Header (2)
- Cookie No HttpOnly Flag (2)
- Cookie without SameSite Attribute (2)
- Server Leaks Version Information via "Powered-By"
- Server Leaks Version Information via "Server"
- X-Content-Type-Options Header Missing (5)
- Authentication Request Identified

Solution:

Upgrade to the latest stable version of PHP, or use the Apache web server and the mod_rewrite module to filter out malicious requests using the "RewriteCond" and "RewriteRule" directives.

Reference:

- <http://projects.webappsec.org/improper-input-handling>
- <http://cwe.mitre.org/data/definitions/89.html>

Alert Tags:

Key	Value
OWASP_2017_A09	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities
WSTG-v42-INPV-12	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing
CVE-2012-1823	https://nvd.nist.gov/vuln/detail/CVE-2012-1823
OWASP_2021_A06	https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/