Executive summary

The 'grey box' strategy was used in a controlled test that Epita Academy administered from August 1 to August 10. This test included internal network addresses and contextual cues with the intention of thoroughly assessing system vulnerabilities.

This exercise's main goal was to carefully examine the server environment, especially user profiles and steganography-related elements. The examination was carried out utilising remote machines that were accessed through the company's VPN server, concentrating on potential prospects for horizontal privilege escalation.

All discovered weaknesses and findings were meticulously exploited and investigated throughout the examination while adhering to predetermined standards. The assessment's findings were thoroughly reported and documented, offering insightful information about the system's security posture.

Approach

Epita academy performed a test under “grey box” method from august 1st to august 10th with some information including the internal network addresses and hints included in order to find test the vulnerability in the system.

The goal was to identify the possible investigation under the server related to user profile and including some information related to steganography.
The test was performed under using remote host using the company VPN server and more towards the horizontal privilege escalation. All the findings and vulnerability was exploited and investigated according to the guideline and reported

Scope

The scope of the assessment is to identify and investigate the remote server to identify the User with other findings throughout the test.

Asset

HOST/URL/IP	DESCRIPTION
10.10.151.0/24	Agent sudo Network THM

Assessment overview and recommendation

During the assessment the academy was able to find major 5 vulnerabilities and TWO login compromises within the system. Since the entire system structure was not implemented security protocols all major findings were based on affecting the system influencing greater threat.

Tester was able to find the weak authentication system in the file server and SSH login. So due another finding is lack of control of implementing strong user password policy by the administrator.

Most of the information were available in clear text and the files were not encrypted or stored using secure format.

Tester also found after the login in to the user account where the major file which included necessary information to further compromise the system and this is high threat to the organization network system and should be fixed with educating the staff and implementing the training to the staff to follow the strong security policies.

Final compromise was the sudo bug in the Server and this routed the tester to compromise the root user without in need for any password.

All the findings were guiding the tester to compromise the system without having to work a lot or research a lot since first the information was clear and system didn't have strong policy to defend any authentication system.

Network Penetration testing Assessment summary

Epita academy started all the necessary testing methodology within the internal network provided by Agent sudo and other information were gathered during the test but wasn't provided by the company such as the information regarding users, configurations and system etc.

Technical finding and severity information

Findings	Severity level	Finding name
1.	HIGH	Clear text information in web page
2.	HIGH	Improper restriction to execesive authentication attempt
3.	Medium	Clear text information available for third party
4.	HIGH	Use of password hashes with less computational effort
5.	HIGH	Authentication bypass using different channel /bug

Summary of findings

Agent sudo compromise walkthrough

The information is to provide that during the test the Epita academy was able to find the user information and to gather evidence of other leakage withing the systems. The technical details will not be included in the walk through but yet will be provided below with proper evidence. The intention of the this gain access to the webserver and also to gain information about the users within the server and also the findings how it led to identify the extra weaknesses in the system.

Compromised detailed walkthrough.

Detailed steps walkthrough with sufficient evidence

After gathering the information of the IP the network nmap scan is performed to identify the open ports and other evidence as the part of information gathering

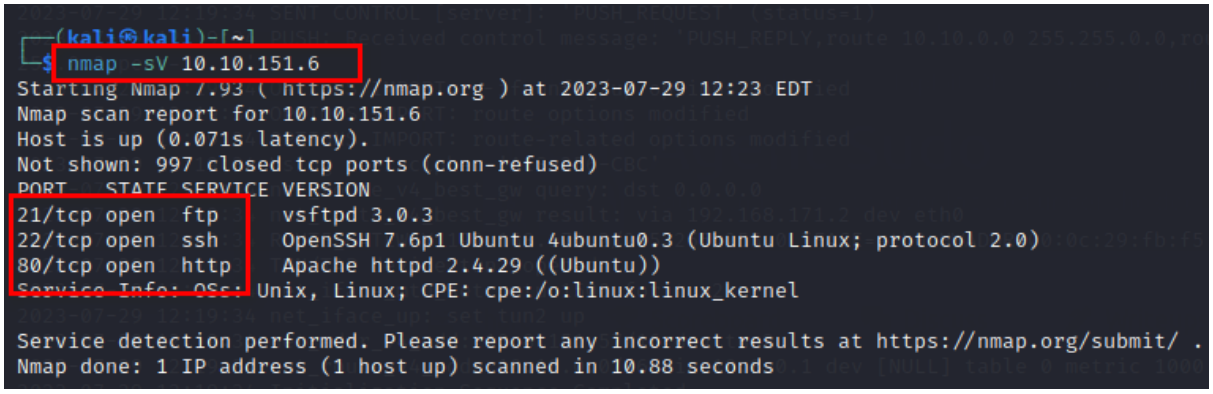


Figure 1. nmap scan finding of open ports

user was able to surface through the web browser and codename access was in “clear text”

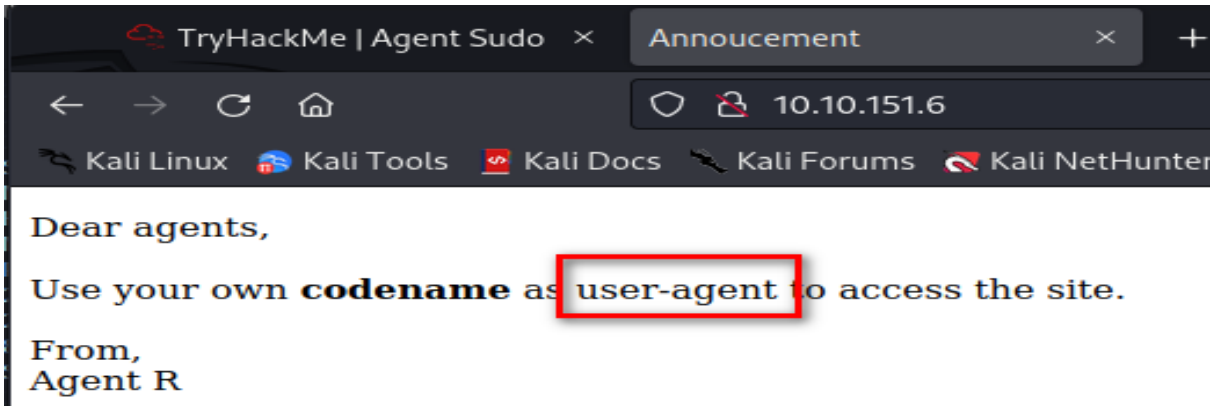


figure 2: web server information in clear text

attempts of sending request to find the user guesses for the access one

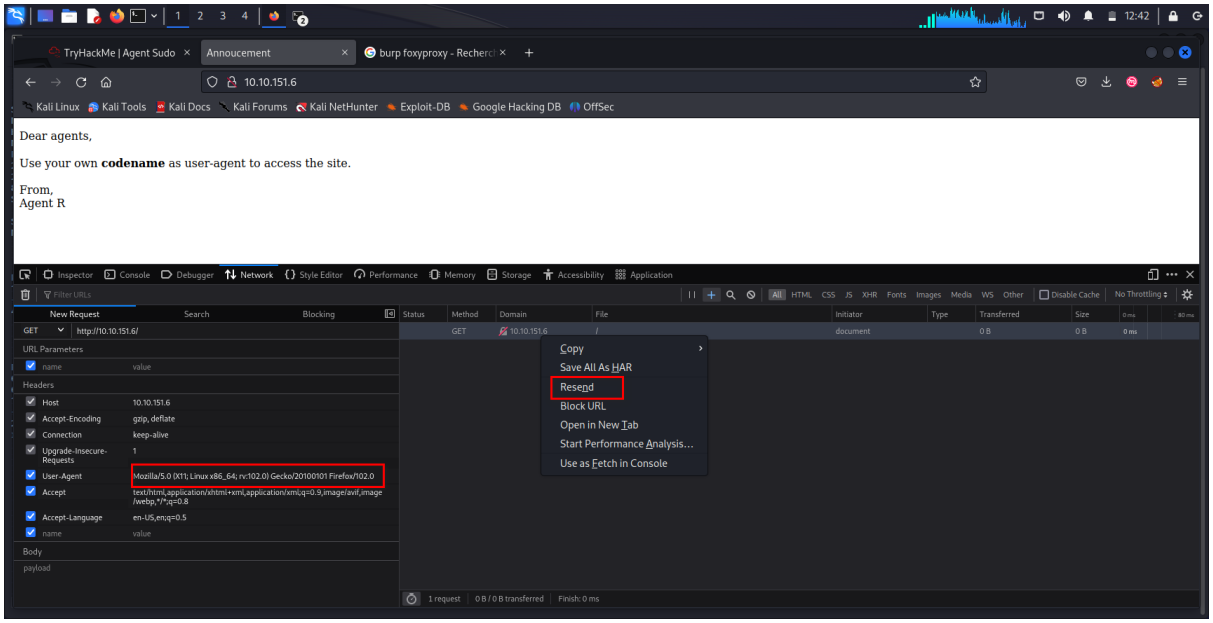


Figure 3: guess one attempt of the username guess

another attempt of multiple times of the request in order to gain the information of user

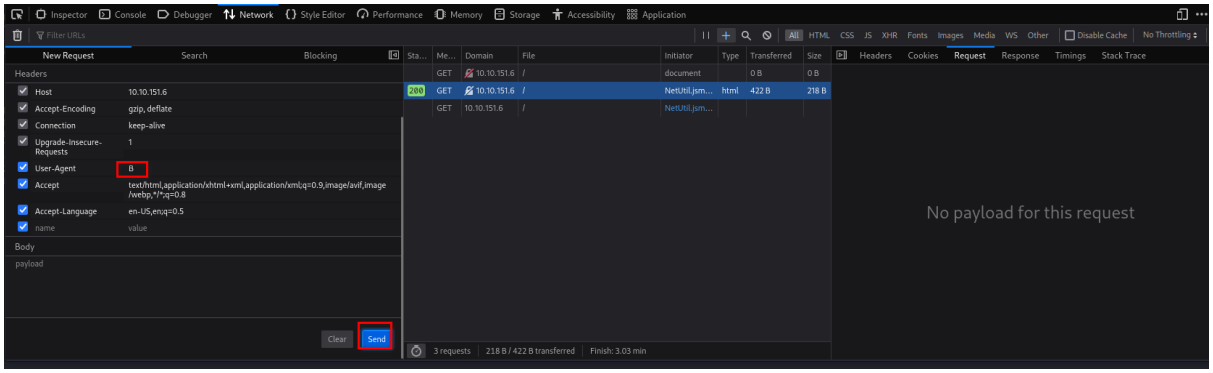


figure 4: guess multiple attempts of user profile

Information access to the user

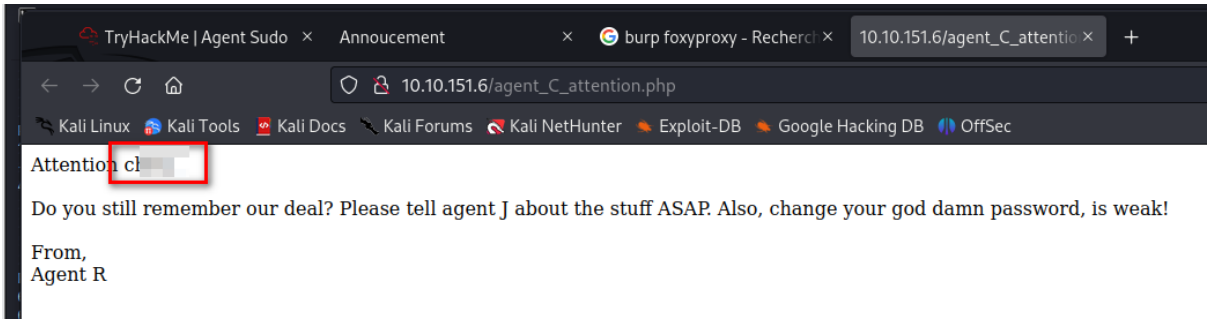


Figure 5: user access seen and information clear text

The attempt of the request to gather the correct page for the user identification

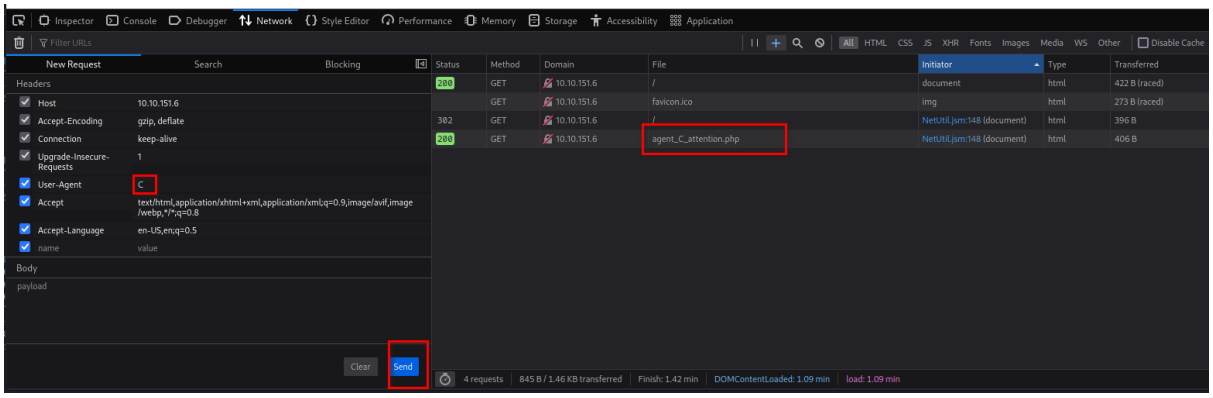


Figure 6: request sent using alphabet C

Now by using the user can brute force to gain the credentials of FTP file server login

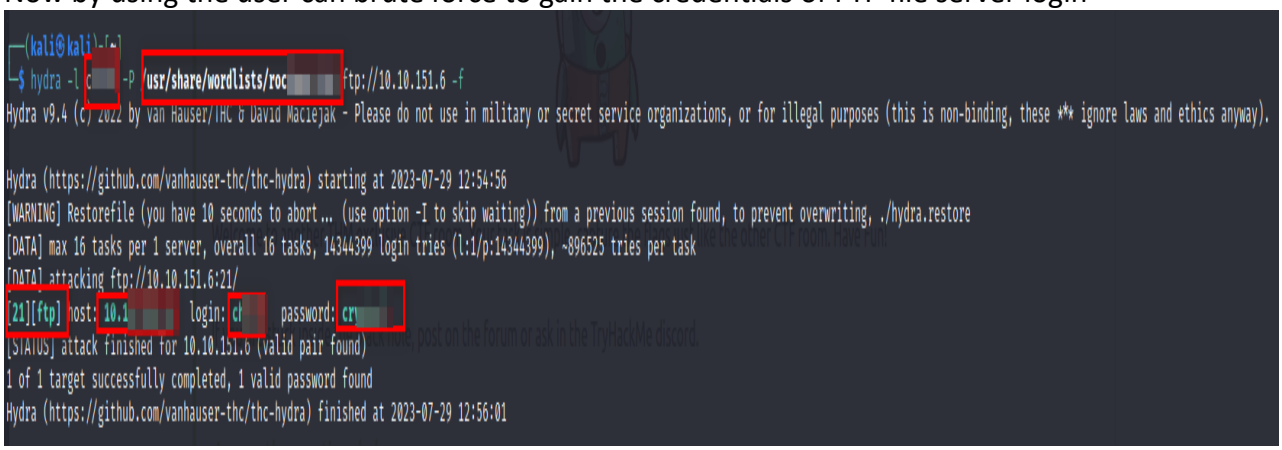


Figure 7: brute forcing using hydra – FTP login credentials

Tester confirmed the login form the bruteforce attack by login into FTP and accessing the files. This shows the vulnerability of the information in clear text and also lack of strong policy to delay the bruteforce authentication attacks

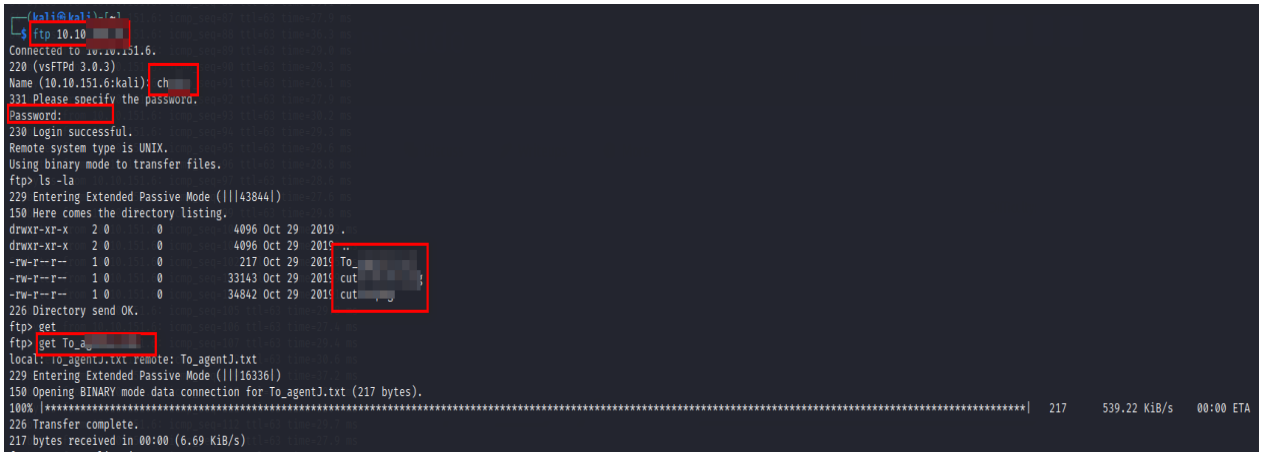


Figure 8: FTP gaining access

Copying and pulling the file after the access

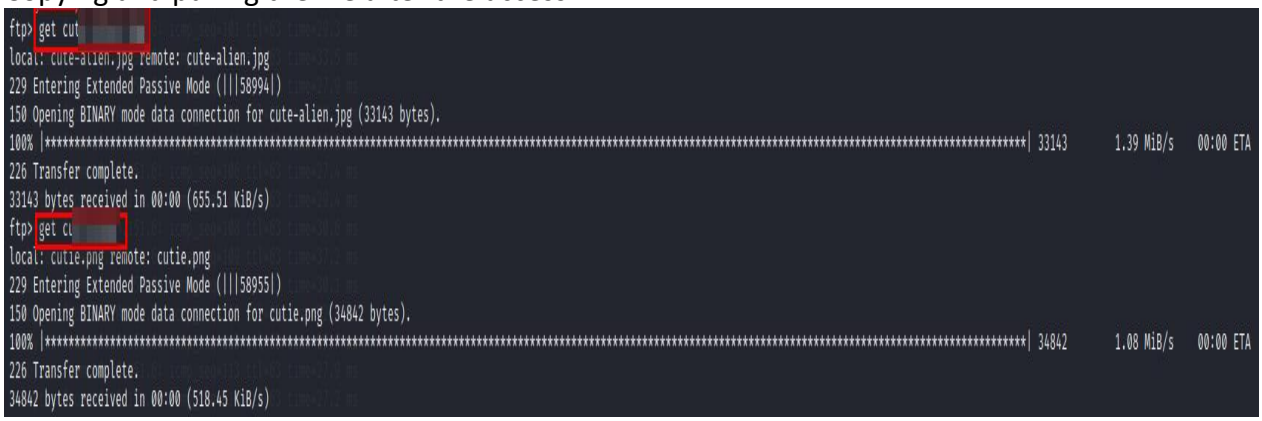


Figure 9: using the access now gaining access to the file

from gained access file the information to the tester was more clear and not encrypted and also created the next pathway to find the vulnerability

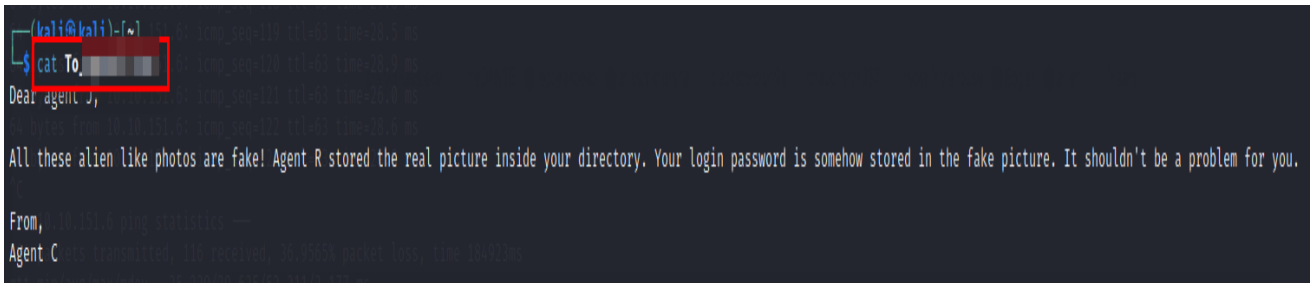


figure 10 : File information

using command line tool in order to gain the information of the image file either for reverse engineer or to crack the hidden information

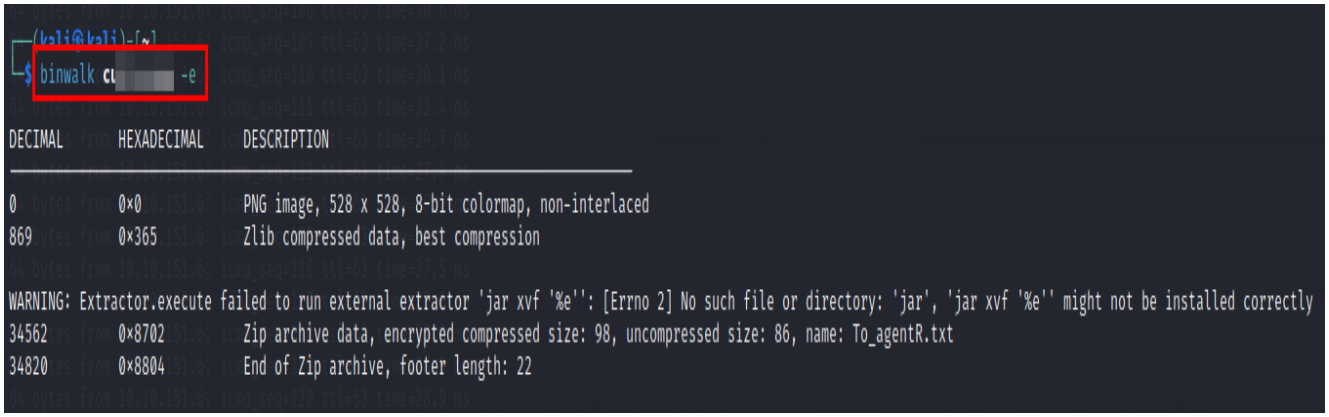


Figure11: binwalk tool to analyze secret file

Analyzing the information file extracted

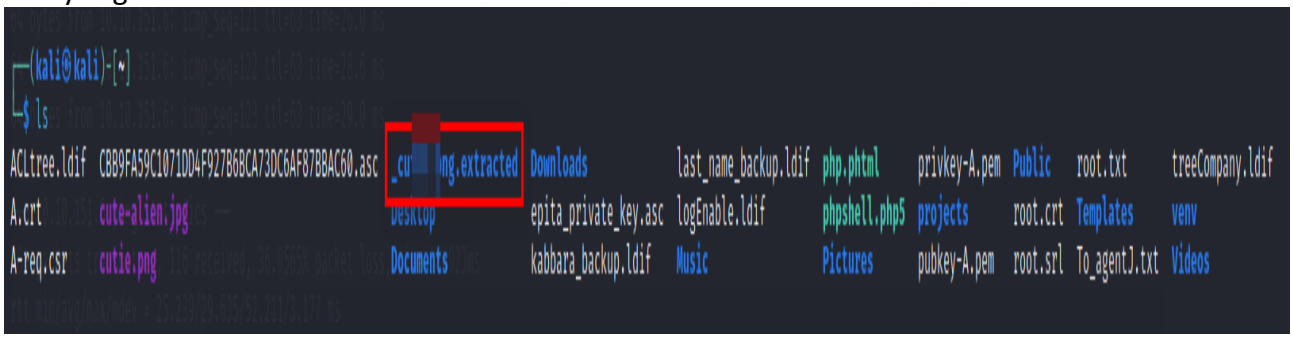


Figure 12: extracted file

Tester confirmed the file location found as the zip file and by using this possibly the next step is to investigate the possible secret within the file

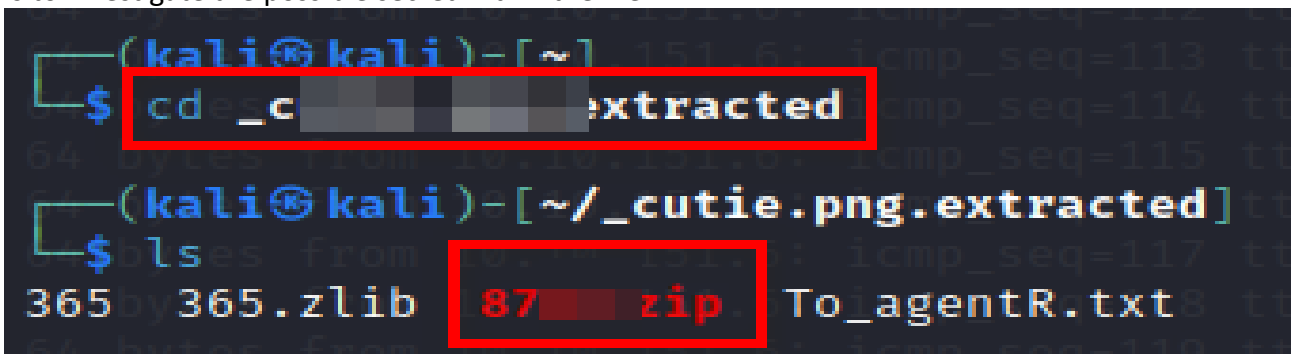


Figure 13: containing zip file from the findings.

Its possible to guess since the information's are form the users standpoint the file can be tested by using tool like john in order to crack the hash out of

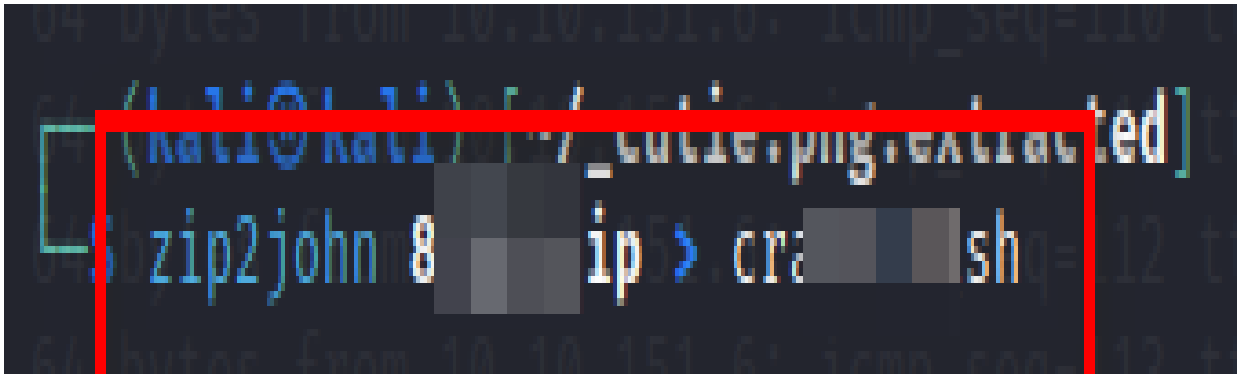


Figure14: using zip2john to gather the hash information.

Using john the tester is capable of redirecting the hashes now using the tool and also the test was successful with receiving the credentials or the passphrase

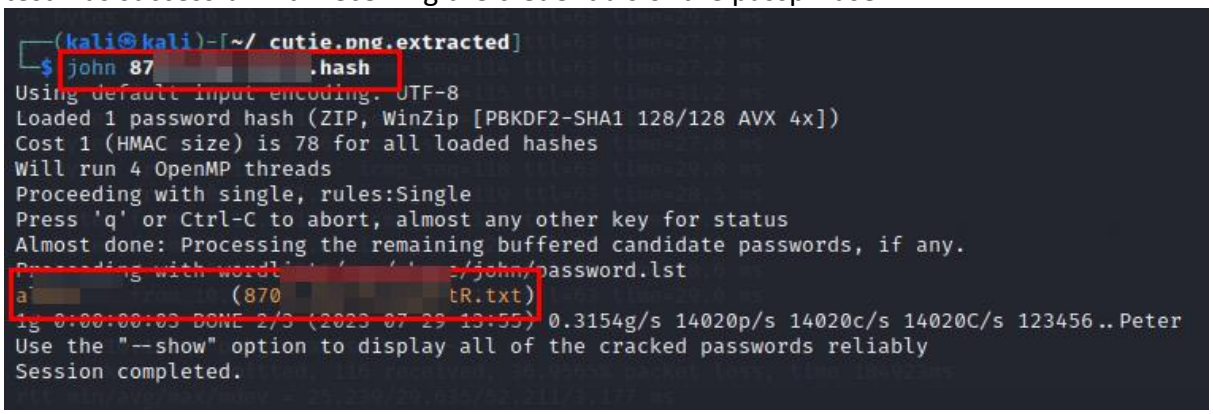


Figure 15: john hash redirecting to catch the passphrase

Using the passprase and using 7zip the attacker was able to unzip the file to gain further message from the user or the hint

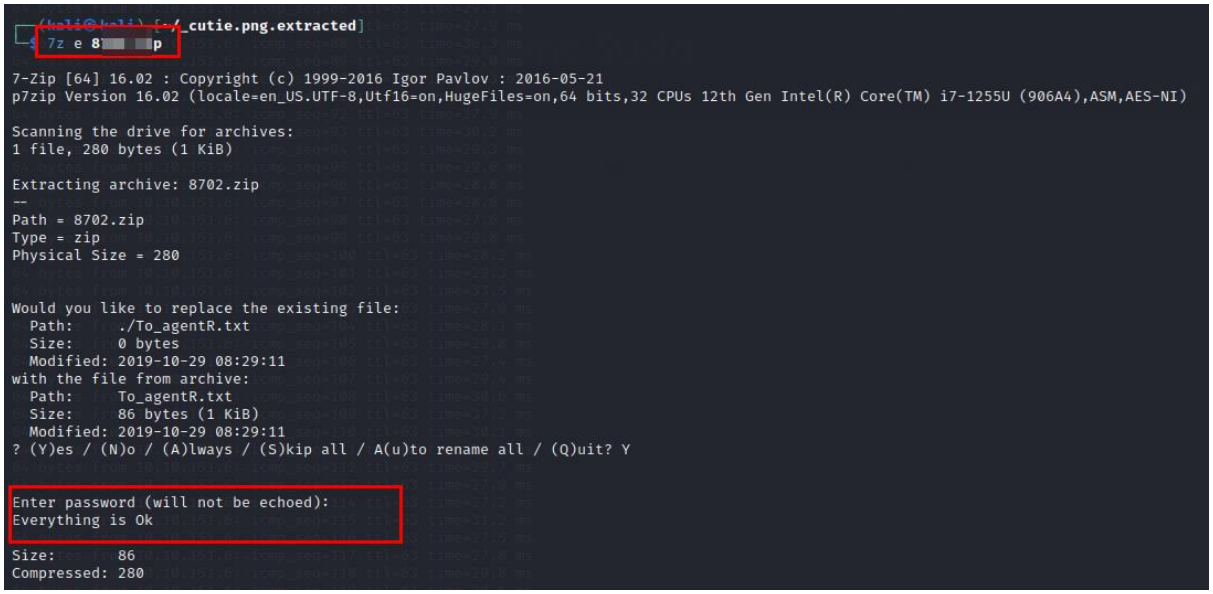


Figure 16: unzipping the file using passphrase

Its more clear to note than again unprotected message within the zip file in order to gain some communication between the user which allows the hacker to be more prone towards targeting more functions towards the process and the message looks like a urgent matter with base 64 typed information

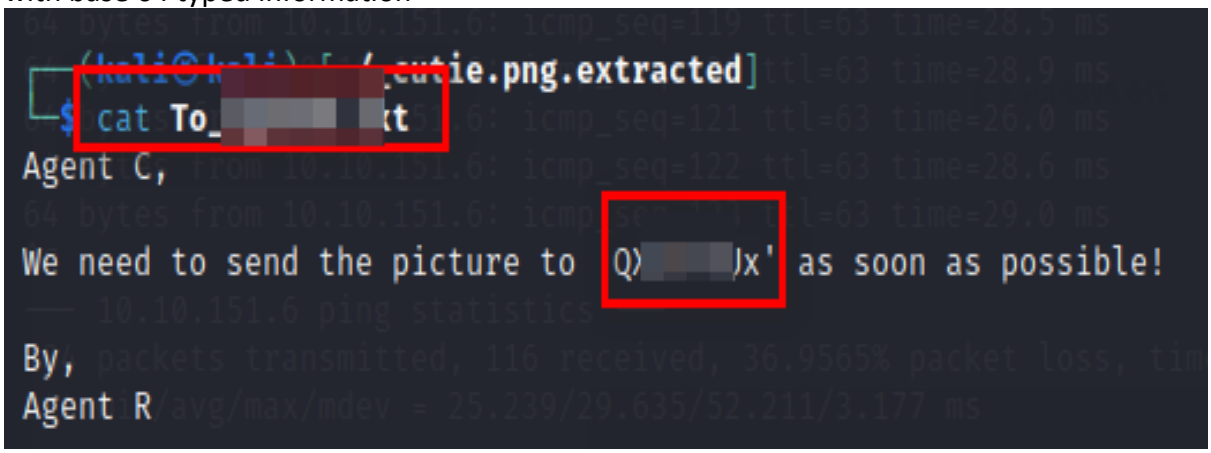


Figure 17: cracked zip file text clear information / message

Now the tester is capable of decoding the base 64 and from that we gained further information of another passphrase

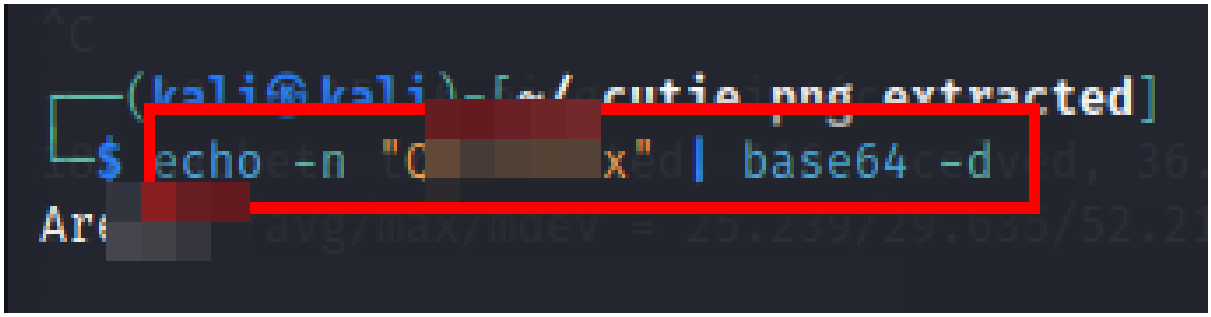


Figure 18: decoding base 64 in order to catch another passphrase

Using steghide and jpg image and also using the passpharase from previous exploit tester is possible to gain another information

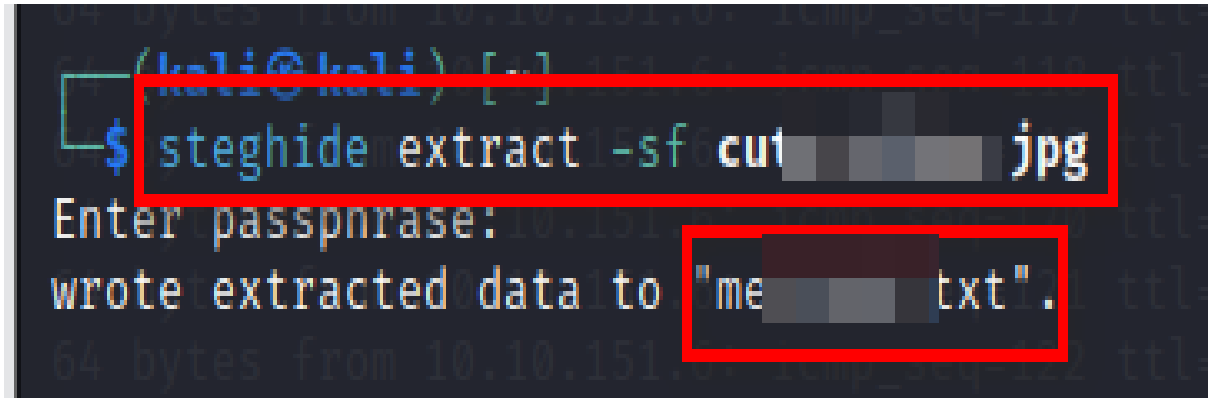


Figure 19: steghide tool helps to extract the message out of image

Using the previous information the attack was able to gain information directly of a password possibly can be used for ssh login and again the authentication method is really weak in the system for the password

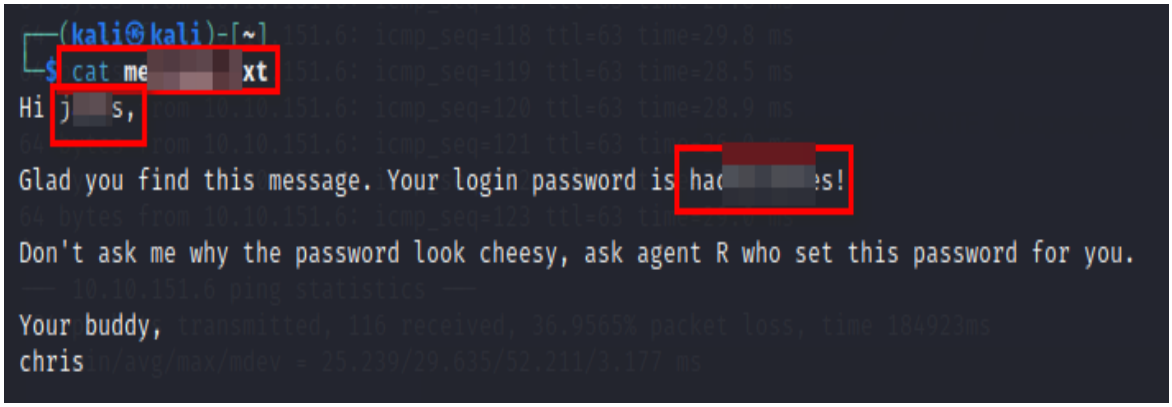


Figure 20: password cracked using cat from a text file

Using the Password the ssh login to the target system is successful and also able to gain the user flags of the system

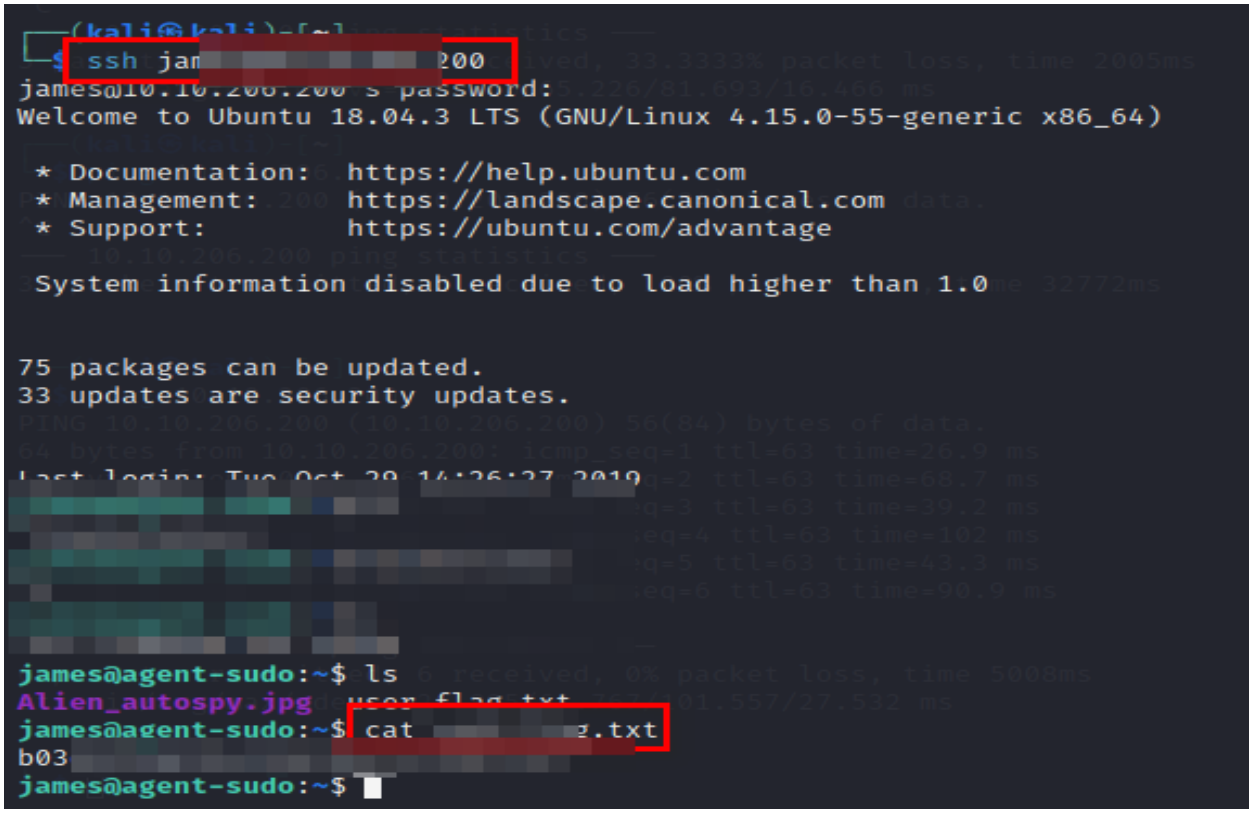


Figure 21: ssh login

Was able to find the sudo bug and also the article accordingly so to gain Root access after the observation

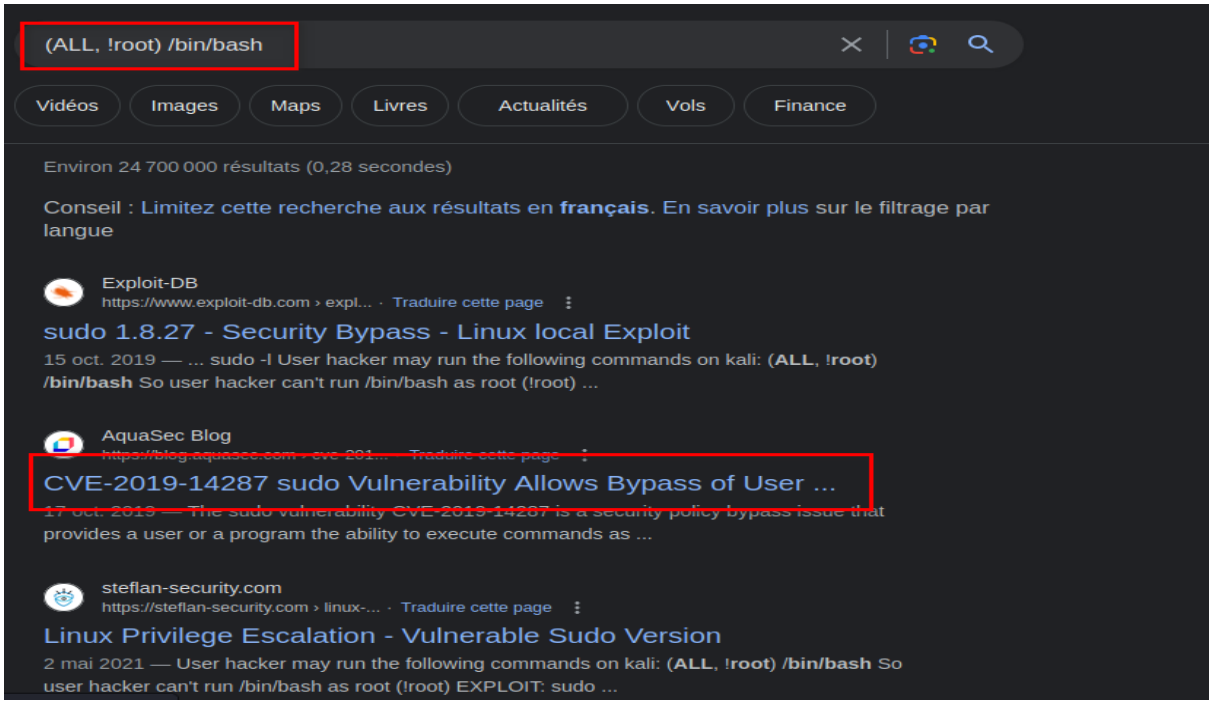


Figure 22 : CVE of the sudo bug vulnerability

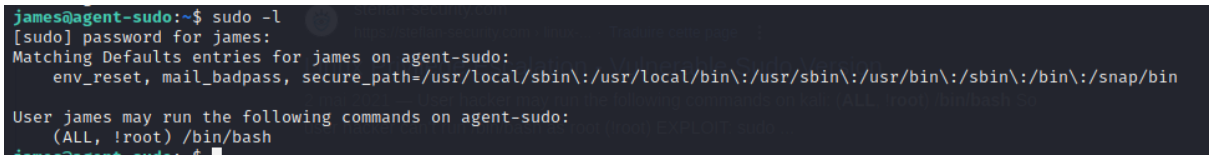


Figure 23: sudo bug information stealing

Using the command sudo bug , tester was able to gain access to the root directory and the flag. the system is totally compromised

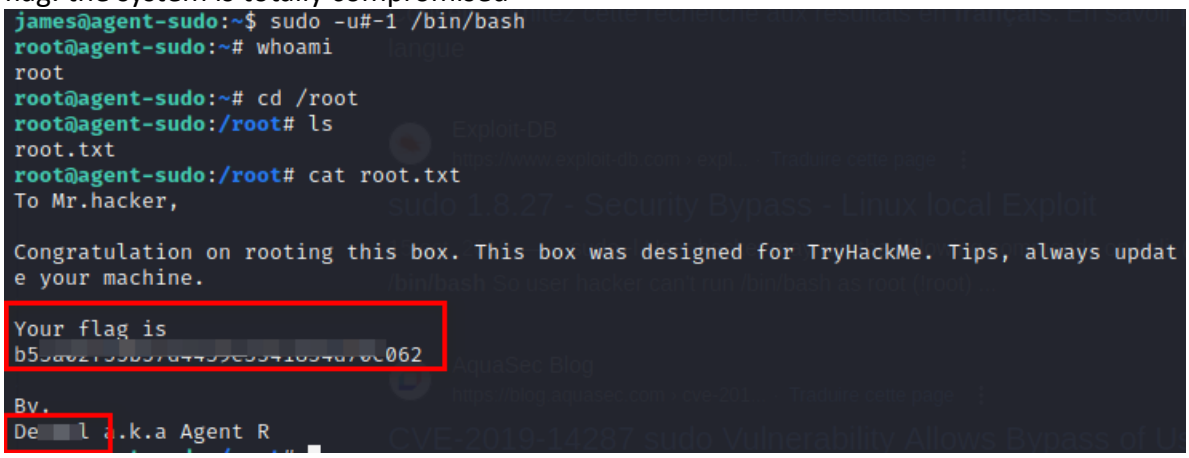


Figure 23. access to the root system using the sudo bug

Remediation summary

As the result from the major findings of this assessment the testers point of view strategies will include three stages of plan accordingly in order to built a remediation technique for increasing the security and eliminating the weak point founded within the system

Short term

- Eliminating the content of secret information in the web page
- Implementing strong password policy for FTP and SSH login
- Using encryption system for the contents displayed or using another platform for these communications.

Medium term

- Implementing view page source content lock
- Using strong administrative security policy for SSH login by creating token authentication
- Adding account lockout for brute force attacks
- Using password complexity to eliminate or increase the timeout for the brute force attacks.

Long term

- Implement regular security audit for password policy and authentication systems.
- Educating the staff and administration to practice the implemented policy for all the authentication system.
- Creating strong background checkup in order to eliminate the important messages leak out within the web server

Technical findings

1. Clear text information displayed in web page of users : High

CWE-319: Cleartext Transmission of Sensitive Information

Weakness ID: 319
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.

Extended Description

Many communication channels can be "sniffed" (monitored) by adversaries during data transmission. For example, in networking, packets can traverse many intermediary nodes from the source to the destination, whether across the internet, an internal network, the cloud, etc. Some actors might have privileged access to a network interface or any link along the channel, such as a router, but they might not be authorized to collect the underlying data. As a result, network traffic could be sniffed by adversaries, spilling security-critical data.

Applicable communication channels are not limited to software products. Applicable channels include hardware-specific technologies such as internal hardware networks and external debug channels, supporting remote JTAG debugging. When mitigations are not applied to combat adversaries within the product's threat model, this weakness significantly lowers the difficulty of exploitation by such adversaries.

When full communications are recorded or logged, such as with a packet dump, an adversary could attempt to obtain the dump long after the transmission has occurred and try to "sniff" the cleartext from the recorded communications in the dump itself. Even if the information is encoded in a way that is not human-readable, certain techniques could determine which encoding is being used, then decode the information.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	311	Missing Encryption of Sensitive Data
ParentOf	V	5	J2EE Misconfiguration: Data Transmission Without Encryption
ParentOf	V	614	Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

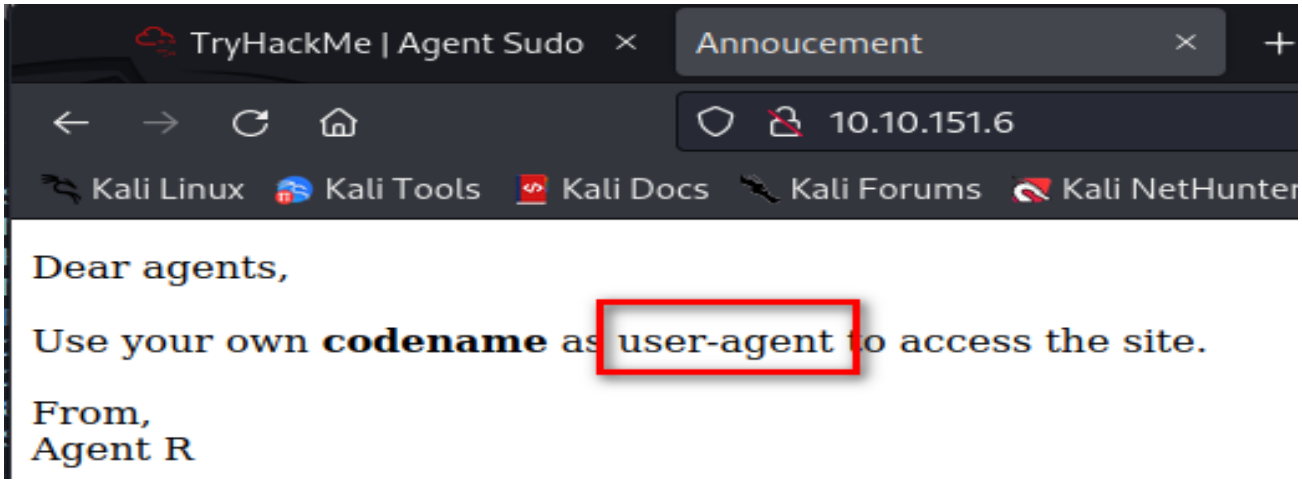
Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	C	199	Information Management Errors

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name
MemberOf	C	1207	Debug and Test Problems

Found evidence : by viewing the web source front page



2. Improper restriction attempts to excessive authentication attempts : High

CWE-307: Improper Restriction of Excessive Authentication Attempts

Weakness ID: 307
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product does not implement sufficient measures to prevent multiple failed authentication attempts within a short time frame, making it more susceptible to brute force attacks.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	799	Improper Control of Interaction Frequency
ChildOf	B	1390	Weak Authentication

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	C	1211	Authentication Errors

Relevant to the view "Weaknesses for Simplified Mapping of Published Vulnerabilities" (CWE-1003)
Relevant to the view "Architectural Concepts" (CWE-1008)

Modes Of Introduction

Phase	Note
Architecture and Design	COMMISSION: This weakness refers to an incorrect design related to an architectural security tactic.

Found evidence : brute force attacks against the FTP authentication

```
┌──(kali㉿kali)-[~]
└─$ hydra -l c[redacted] -P /usr/share/wordlists/roc[redacted] ftp://10.10.151.6 -f
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 2023-07-29 12:54:56
[WARNING] Restorefile (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, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ftp://10.10.151.6:21/
[21][ftp] host: 10.1[redacted]   login: c[redacted]   password: cr[redacted]
[STATUS] attack finished for 10.10.151.6 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-07-29 12:56:01
```

3. Clear text information available for third party : Medium

CWE-312: Cleartext Storage of Sensitive Information

Weakness ID: 312
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product stores sensitive information in cleartext within a resource that might be accessible to another control sphere.

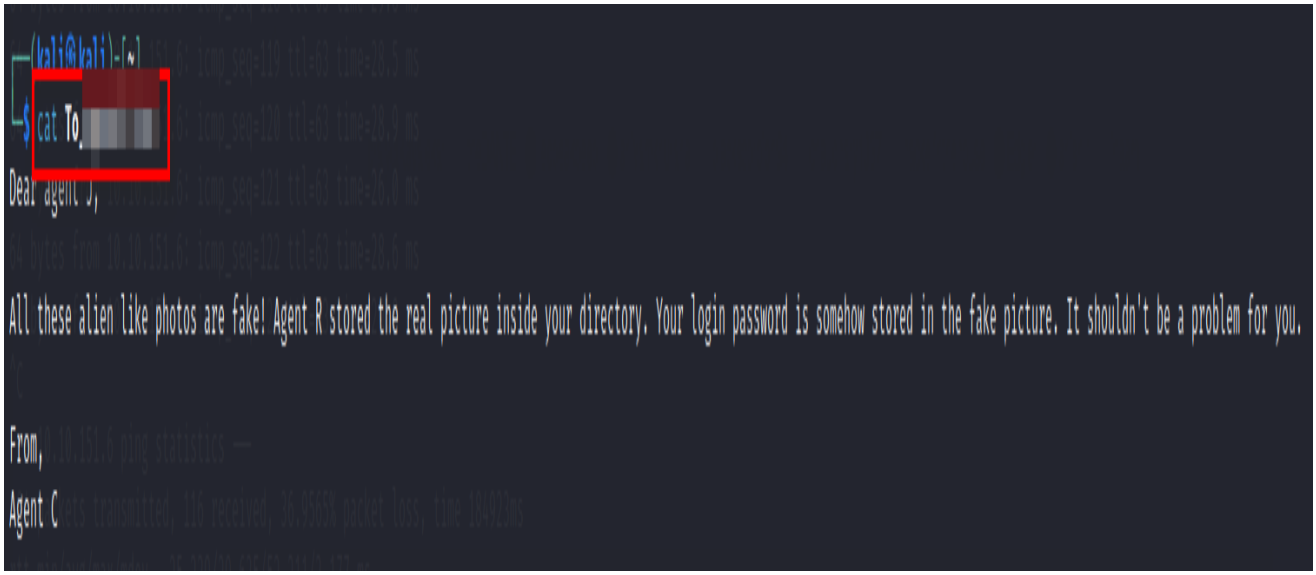
Extended Description

Because the information is stored in cleartext (i.e., unencrypted), attackers could potentially read it. Even if the information is encoded in a way that is not human-readable, certain techniques could determine which encoding is being used, then decode the information.

When organizations adopt cloud services, it can be easier for attackers to access the data from anywhere on the Internet.

In some systems/environments such as cloud, the use of "double encryption" (at both the software and hardware layer) might be required, and the developer might be solely responsible for both layers, instead of shared responsibility with the administrator of the broader system/environment.

Found evidence : the secret information and clues were found



4. Use of password hashes with less computational effort : High

CWE-916: Use of Password Hash With Insufficient Computational Effort

Weakness ID: 916
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | Complete | Custom

Description

The product generates a hash for a password, but it uses a scheme that does not provide a sufficient level of computational effort that would make password cracking attacks infeasible or expensive.

Extended Description

Many password storage mechanisms compute a hash and store the hash, instead of storing the original password in plaintext. In this design, authentication involves accepting an incoming password, computing its hash, and comparing it to the stored hash.

Many hash algorithms are designed to execute quickly with minimal overhead, even cryptographic hashes. However, this efficiency is a problem for password storage, because it can reduce an attacker's workload for brute-force password cracking. If an attacker can obtain the hashes through some other method (such as SQL injection on a database that stores hashes), then the attacker can store the hashes offline and use various techniques to crack the passwords by computing hashes efficiently. Without a built-in workload, modern attacks can compute large numbers of hashes, or even exhaust the entire space of all possible passwords, within a very short amount of time, using massively-parallel computing (such as cloud computing) and GPU, ASIC, or FPGA hardware. In such a scenario, an efficient hash algorithm helps the attacker.

Found evidence : using john were able to crack the hashes easily

```
┌──(kali㉿kali)-[~/ cutie.png.extracted]
└─$ john 87[redacted].hash
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 AVX 4x])
Cost 1 (HMAC size) is 78 for all loaded hashes
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
[redacted]password.lst
a[redacted] (870[redacted]tR.txt)
[redacted] 0.3154g/s 14020p/s 14020c/s 14020C/s 123456..Peter
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

5. Authentication bypass using different bug: High

CWE-288: Authentication Bypass Using an Alternate Path or Channel

Weakness ID: 288
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

A product requires authentication, but the product has an alternate path or channel that does not require authentication.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	306	Missing Authentication for Critical Function
ParentOf	B	425	Direct Request ('Forced Browsing')
ParentOf	B	1299	Missing Protection Mechanism for Alternate Hardware Interface
PeerOf	B	420	Unprotected Alternate Channel

Relevant to the view "Architectural Concepts" (CWE-1008)
Relevant to the view "CISQ Data Protection Measures" (CWE-1340)

Modes Of Introduction

Phase	Note
Architecture and Design	COMMISSION: This weakness refers to an incorrect design related to an architectural security tactic.
Architecture and Design	This is often seen in web applications that assume that access to a particular CGI program can only be obtained through a "front" screen, when the supporting programs are directly accessible. But this problem is not just in web apps.

Applicable Platforms

Languages

Class: Not Language-Specific *(Undetermined Prevalence)*

Common Consequences

Scope	Impact	Likelihood
Access Control	**Technical Impact:** *Bypass Protection Mechanism*	

Found evidence : bypassing the user root system bypassing the password