# Startup CTF Writeup

1. **Reconnissance**

Nmap scan:

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

As we can see, there are three open ports on the system. The major issue is that the FTP server allows anonymous login.

FTP Anonymous Login: FTP (File Transfer Protocol) anonymous login refers to a method of accessing an FTP server without providing explicit credentials such as a username or password. In an anonymous login scenario, users can log in to the FTP server with the default username "anonymous" or "ftp" and use their email address as the password or provide any random string as a password.

This method is often used for public FTP servers that allow users to download or upload files without requiring individual user accounts. While it provides convenient access, users should be cautious when using anonymous login to avoid potential security risks. Server administrators may choose to restrict certain actions or implement additional security measures to mitigate potential abuse.

After the Nmap scan, I attempted to log in to the FTP server. In this scenario, due to the system allowing anonymous login, I gained access to the FTP server.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

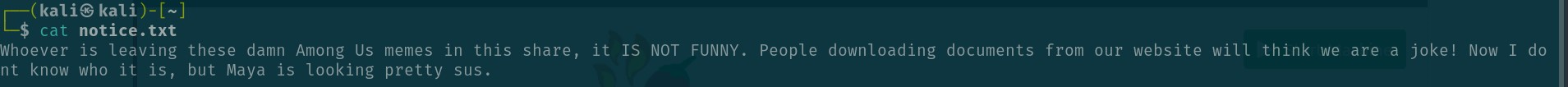
After gaining access, I downloaded all files from the FTP server.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

çizgi film, ekran görüntüsü, metin, memeli içeren bir resim

Açıklama otomatik olarak oluşturuldu



These files are not useful in this case. However, I noticed that I should check the web page.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

I conducted directory fuzzing within the system during this phase

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Allowing GUI access to the FTP server directly from the web server introduces security risks that should be avoided. However, in this case, the victim was not aware of this situation.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

After reaching a dead end, I attempted to upload a PHP reverse shell to the system. Due to FTP anonymous login being enabled, the chances of success were high.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Allowing users to upload any file in FTP servers may cause serious problems. In this case, I used a PHP shell to obtain a reverse shell, but many attacks can be performed in this situation in real life. After triggering the PHP file in the web browser, I obtained a reverse shell.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

I stabilized the shell and obtained the initial access.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

As we can see, we don't have many privileges in this account. In most cases, service accounts don't have access to the system. If we look around, we can see that there is an 'incident' directory in the file system

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

I retrieved the PCAP file to my local machine. This is also a serious problem. The company should not allow critical data exfiltration. I checked the PCAP file and noticed that someone stole a user password.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this case, the company should not store this kind of sensitive document in an easily accessible area. I decided to attempt the password on the SSH server..metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Another serious problem is that after the password leak, users did not change their passwords metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Documents do not contain any sensitive data. I checked the scripts directorymetin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

When I examined the 'print.sh' file, I noticed that I had writing rights. I injected an NC shell into the script. After triggering 'planner.sh,' I obtained a root shell.metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

# **Suggestions**

A ) File Upload

**Uploading executable files like PHP to an FTP server can pose significant security risks due to the following reasons:**

1. **Code Execution Vulnerability:**
   * Allowing the upload of executable files introduces the risk of code execution on the server. If an attacker successfully uploads a malicious PHP file, they can potentially run arbitrary code, leading to unauthorized access, data breaches, and other security compromises.
2. **Remote Code Execution (RCE):**
   * PHP files, when executed on a server, can enable remote code execution. Attackers may exploit vulnerabilities in the uploaded PHP scripts to execute commands on the server, giving them unauthorized control over the system.
3. **Web Shell Exploitation:**
   * Uploading PHP files could be a means for attackers to deploy web shells on the server. Web shells provide a backdoor for malicious activities, allowing unauthorized users to interact with the server, manipulate files, and execute commands remotely.
4. **Security Bypass and Elevation of Privileges:**
   * Malicious PHP files may contain code designed to exploit vulnerabilities in the server's security mechanisms. This could result in the bypass of access controls, elevation of privileges, and unauthorized manipulation of sensitive data.
5. **Denial of Service (DoS) Attacks:**
   * By uploading PHP files that consume excessive server resources, an attacker can orchestrate denial-of-service attacks. This can lead to the unavailability of services, disrupting the normal functioning of the server and affecting legitimate users.
6. **Injection Attacks:**
   * If the FTP server allows the execution of uploaded PHP files, it may be susceptible to injection attacks. Attackers could inject malicious code into the uploaded PHP files, leading to SQL injection, Cross-Site Scripting (XSS), or other types of injection vulnerabilities.

**Mitigation Measures:**

* Implement strict file upload policies, allowing only specific file types and restricting executable content.
* Regularly update and patch the FTP server software to address known vulnerabilities.
* Conduct security audits to identify and remediate potential weaknesses in the server's configuration.
* Utilize firewalls and intrusion detection/prevention systems to monitor and block suspicious activities.

In summary, permitting the upload of executable files like PHP to an FTP server exposes it to various security threats, including code execution, remote code execution, and unauthorized access, necessitating robust security measures to mitigate these risks.

B ) Perform security hardening operations on all users without exceptions. If a file has execute rights, make sure that least privileged users cannot modify it.

**Mitigation Suggestions:**

1. **Implement Strong Authentication:**
   * Enforce the use of strong authentication mechanisms, such as multi-factor authentication, to enhance user account security.
2. **Regular Password Policies:**
   * Establish and enforce regular password change policies to mitigate the impact of potential password leaks.
3. **User Education:**
   * Conduct user awareness training to educate users on the importance of changing passwords after security incidents, emphasizing the need for proactive security measures.
4. **Automated Password Expiry Notifications:**
   * Implement automated notifications to prompt users to change their passwords regularly, reducing the likelihood of overlooking password changes.
5. **Security Hardening Best Practices:**
   * Follow security hardening best practices for user accounts, including the principle of least privilege, to limit user access to only the necessary resources.
6. **File Permissions Review:**
   * Regularly review and update file permissions, especially for files with execute rights. Ensure that least privileged users cannot modify critical executable files.
7. **Monitoring and Auditing:**
   * Implement robust monitoring and auditing mechanisms to detect and alert on unauthorized access or modifications to files. Regularly review audit logs for security incidents.
8. **Automated Security Scans:**
   * Utilize automated security scanning tools to identify and remediate vulnerabilities in user accounts and file permissions.
9. **Access Control Lists (ACLs):**
   * Use Access Control Lists (ACLs) to fine-tune file permissions, restricting modification rights for users who do not require such access.
10. **Regular Security Assessments:**
    * Conduct regular security assessments, including penetration testing and vulnerability scanning, to identify and address potential weaknesses in user account security and file permissions.

Implementing these mitigation strategies can enhance overall security posture and minimize the risk associated with password leaks and unauthorized modifications to files.