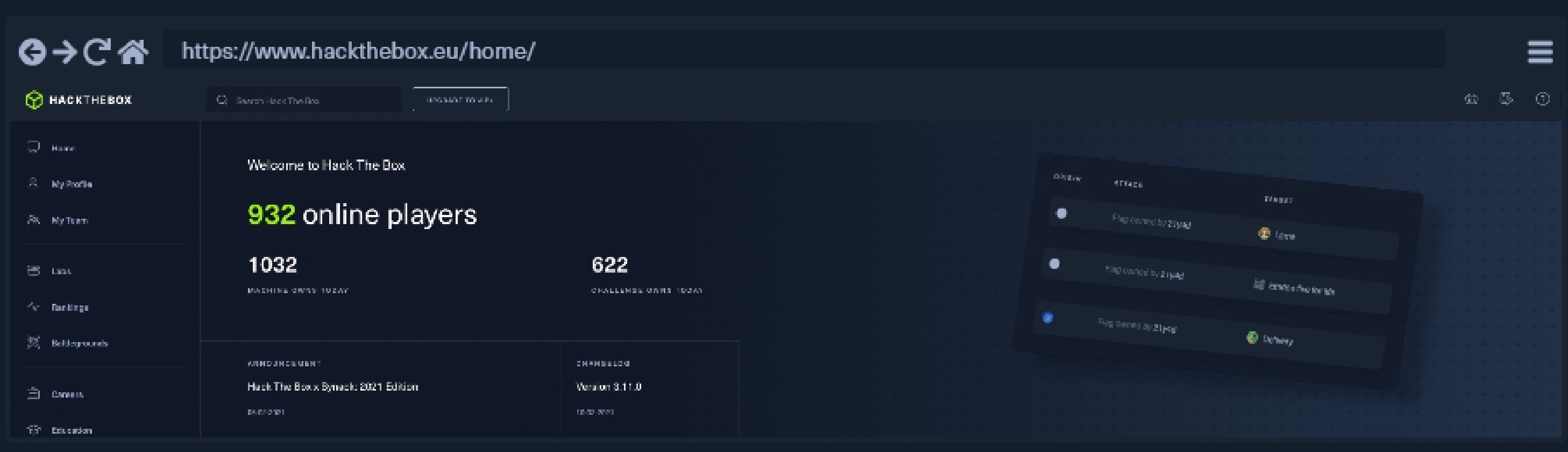
Introduction

Web applications are interactive applications that run on web browsers. Web applications usually adopt a client-server architecture to run and handle interactions. They typically have front end components (i.e., the website interface, or "what the user sees") that run on the client-side (browser) and other back end components (web application source code) that run on the server-side (back end server/databases).

This allows organizations to host powerful applications with near-complete real-time control over their design and functionality while being accessible worldwide. Some examples of typical web applications include online email services like `Gmail`, online retailers `Amazon`, and online word processors like `Google Docs`.



https://www.hackthebox.eu/home/

Welcome to Hack The Box

932 online players

1032

622

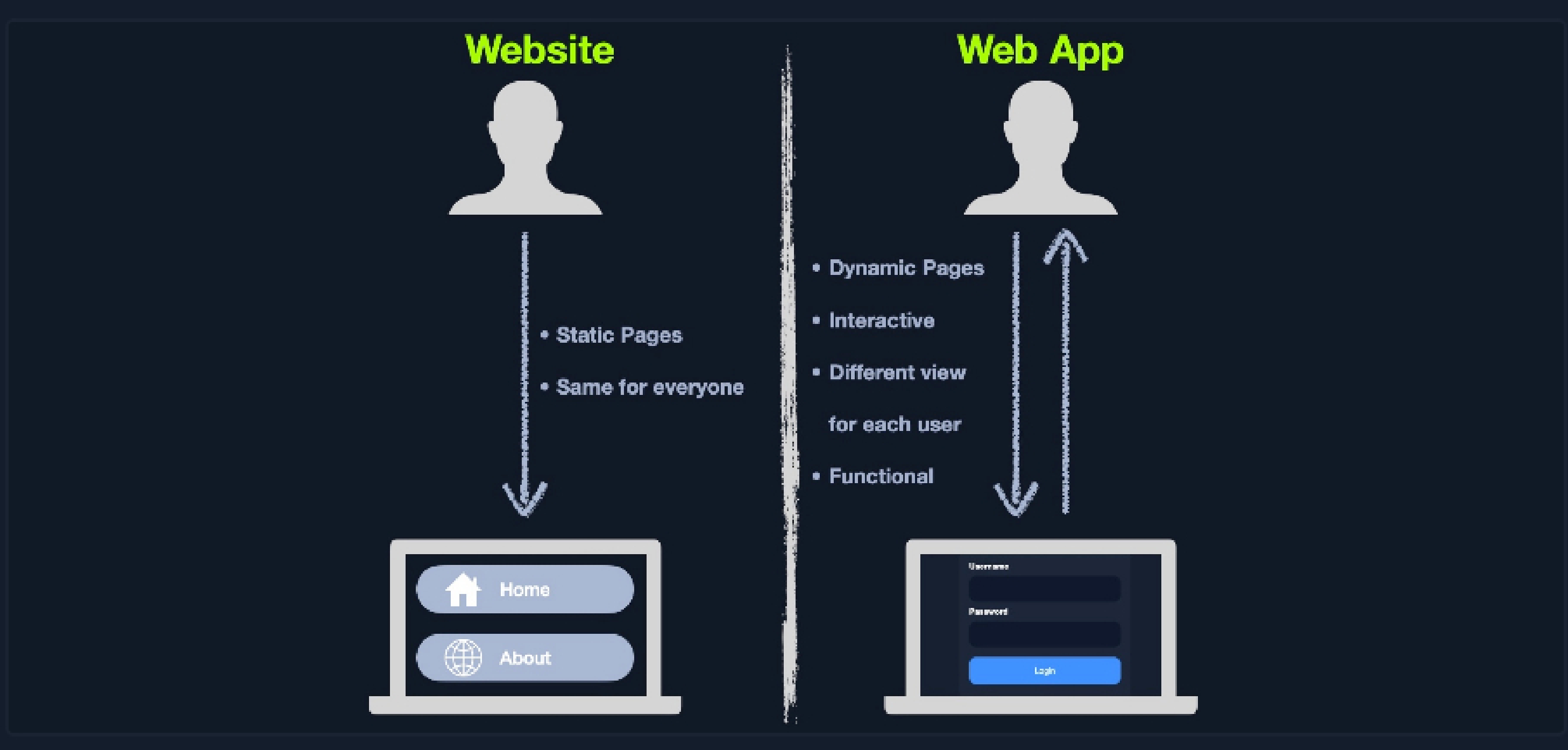
Hack The Box x Synack: 2021 Edition

Version 3.11.0

Web applications are not exclusive to giant providers like Google or Microsoft but can be developed by any web developer and hosted online in any of the common hosting services, to be used by anyone on the internet. This is why today we have millions of web applications all over the internet, with billions of users interacting with them every day.

Web Applications vs. Websites

In the past, we interacted with websites that are static and cannot be changed in real-time. This means that traditional websites were statically created to represent specific information, and this information would not change with our interaction. To change the website's content, the corresponding page has to be edited by the developers manually. These types of static pages do not contain functions and, therefore, do not produce real-time changes. That type of website is also known as Web 1.0.



On the other hand, most websites run web applications, or Web 2.0 presenting dynamic content based on user interaction. Another significant difference is that web applications are fully functional and can perform various functionalities for the end-user, while web sites lack this type of functionality. Other key differences between traditional websites and web applications include: Being modular Running on any display size Running on any platform without being optimized

Web Applications vs. Native Operating System Applications

Unlike native operating system (native OS) applications, web applications are platform-independent and can run in a browser on any operating system. These web applications do not have to be installed on a user's system because these web applications and their functionality are executed remotely on the remote server and hence do not consume any space on the end user's hard drive.

Another advantage of web applications over native OS applications is version unity. All users accessing a web application use the same version and the same web application, which can be continuously updated and modified without pushing updates to each user. Web applications can be updated in a single location (webserver) without developing different builds for each platform, which dramatically reduces maintenance and support costs removing the need to communicate changes to all users individually.

On the other hand, native OS applications have certain advantages over web applications, mainly their operation speed and the ability to utilize native operating system libraries and local hardware. As native applications are built to utilize native OS libraries, they are much faster to load and interact with. Furthermore, native applications are usually more capable than web applications, as they have a deeper integration to the operating system and are not limited to the browser's capabilities only.

More recently, however, hybrid and progressive web applications are becoming more common. They utilize modern frameworks to run web applications using native OS capabilities and resources, making them faster than regular web applications and more capable.

Web Application Distribution

There are many open-source web applications used by organizations worldwide that can be customized to meet each organization's needs. Some common open source web applications include:

- `WordPress`
- `OpenCart`
- `Joomla`

There are also proprietary 'closed source' web applications, which are usually developed by a certain organization and then sold to another organization or used by organizations through a subscription plan model. Some common closed source web applications include:

- `Wix`
- `Shopify`
- `DotNetNuke`

Security Risks of Web Applications

Web application attacks are prevalent and present a challenge for most organizations with a web presence, regardless of their size. After all, they are usually accessible from any country by everyone with an internet connection and a web browser and usually offer a vast attack surface. There are many automated tools for scanning and attacking web applications that, in the wrong hands, can cause significant damage. As web applications become more complicated and advanced, so does the possibility of critical vulnerabilities being incorporated into their design.

A successful web application attack can lead to significant losses and massive business interruptions. Since web applications are run on servers that may host other sensitive information and are often also linked to databases containing sensitive user or corporate data, all of this data could be compromised if a web site is successfully attacked. This is why it is critical for any business that utilizes web applications to properly test these applications for vulnerabilities and patch them promptly while testing that the patch fixes the flaw and does not inadvertently introduce any new flaws.

Web application penetration testing is an increasingly critical skill to learn. Any organization looking to secure their internet-facing (and internal) web applications should undergo frequent web application tests and implement secure coding practices at every development life cycle stage. To properly pentest web applications, we need to understand how they work, how they are developed, and what kind of risk lies at each layer and component of the application depending on the technologies in use.

We will always come across various web applications that are designed and configured differently. One of the most current and widely used methods for testing web applications is the OWASP Web Security Testing Guide.

One of the most common procedures is to start by reviewing a web application's front end components, such as `HTML`, `CSS` and `JavaScript` (also known as the front end trinity), and attempt to find vulnerabilities such as Sensitive Data Exposure and Cross-Site Scripting (XSS). Once all front end components are thoroughly tested, we would typically review the web application's core functionality and the interaction between the browser and the webserver to enumerate the technologies the webserver uses and look for exploitable flaws. We typically assess web applications from both an unauthenticated and authenticated perspective (if the application has login functionality) to maximize coverage and review every possible attack scenario.

Attacking Web Applications

In this day and age most every company, no matter the size has one or more web applications within their external perimeter. These applications can be everything from simple static websites to blogs powered by Content Management Systems (CMS) such as `WordPress` to complicated applications with sign-up/login functionality supporting various user roles from basic users to super admins. Nowadays, it is not very common to find an externally facing host directly exploitable via a known public exploit (such as a vulnerable service or Windows remote code execution (RCE) vulnerability), though it does happen. Web applications provide a vast attack surface, and their dynamic nature means that they are constantly changing (and overlooked!). A relatively simple code change may introduce a catastrophic vulnerability or a series of vulnerabilities that can be chained together to gain access to sensitive data or remote code execution on the webserver or other hosts in the environment, such as database servers.

In my experience, it is not uncommon to find flaws that can lead directly to code execution, such as a file upload form that allows for the upload of malicious code or a file inclusion vulnerability that can be leveraged to obtain remote code execution. A well-known vulnerability that is still quite prevalent in various types of web applications is SQL injection. This type of vulnerability arises from the unsafe handling of user-supplied input. It can result in access to sensitive data, reading/writing files on the database server, and even remote code execution.

We often find SQL injection vulnerabilities on web applications that use Active Directory for authentication. While we can usually not leverage this to extract passwords (since Active Directory administers them), we can often pull most or all Active Directory user email addresses, which are often the same as their usernames. This data can then be used to perform a password spraying attack against web portals that use Active Directory for authentication such as VPN or Microsoft Outlook Web Access/Microsoft O365. A successful password spray can often result in access to sensitive data such as email or even a foothold directly into the corporate network environment.

This example shows the damage that can arise from a single web application vulnerability, especially when "chained" to extract data from one application that can be used to attack other portions of a company's external infrastructure. A well-rounded infosec professional should have a deep understanding of web applications and be as comfortable attacking web applications as performing network penetration testing and Active Directory attacks. A penetration tester with a strong foundation in web applications can often set themselves apart from their peers and find flaws that others may overlook. A few more real-world examples of web application attacks and the impact are as follows:

Flaw	Real-world Scenario
SQL injection	Obtaining Active Directory usernames and performing a password spraying attack against a VPN or email portal.
File Inclusion	Reading source code to find a hidden page or directory which exposes additional functionality that can be used to gain remote code execution.
Unrestricted File Upload	A web application that allows a user to upload a profile picture that allows any file type to be uploaded (not just images). This can be leveraged to gain full control of the web application server by uploading malicious code.
Insecure Direct Object Referencing (IDOR)	When combined with a flaw such as broken access control, this can often be used to access another user's files or functionality. An example would be editing your user profile browsing to a page such as /user/ 701/edit-profile. If we can change the `701` to `702`, we may edit another user's profile!
Broken Access Control	Another example is an application that allows a user to register a new account. If the account registration functionality is designed poorly, a user may perform privilege escalation when registering. Consider the `POST` request when registering a new user, which submits the data `username=bjones&password=Welcome1&email=bjones@inlanefreight.local&roleid=3`. What if we can manipulate the `roleid` parameter and change it to `0` or `1`. We have seen real-world applications where this was the case, and it was possible to quickly register an admin user and access many unintended features of the web application.

Start becoming familiar with common web application attacks and their implications. Don't worry if any of these terms sound foreign at this point; they will become clearer as you progress and apply an iterative approach to learning.

It is imperative to study web applications in-depth and become familiar with how they work and many different application stacks. We will see web application attacks repeatedly during our Academy journey, on the main HTB platform, and in real-life assessments. Let's dive in and learn the structure/function of web applications to become better-informed attackers, set us apart from our peers, and find flaws that others may overlook. It is imperative to study web applications in-depth and become familiar with how they work and many different application stacks and types of applications.

Next ➔

Mark Complete & Next

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left