

Making Hacking Accessible

Let's talk about making hacking accessible!

October 1, 2020 · 20 min · Bee

I hate the current state of hacking education, or of hacking tools. None of it is accessible to any minority.

In this post, I aim to distil some guidelines for making hacking accessible.

I am by no means an expert. I don't even work in A11Y. However, I do have some wonderful friends that do — and I am passionate about this subject.

Why We Need to Focus on Accessibility

1. Basic human rights
2. An expansion in the infosec community of new tools, new learning platforms, and new hackers
3. SEO
4. It is legally required.
5. It makes the experience better for everyone.

Let's go through these.

Basic Human Rights

Hacking isn't just inaccessible, it is the opposite — it is actively discluding members from the community because of some issues they were born or developed that they cannot help.

It is a basic human right to extend everything we do to be accessible to everyone. In the same way, it is a right for you to get healthcare, to get an education.

Around 15% of all people suffer from some sort of disability. See this blog post for more info on why accessibility is important or the world report on disability for statistics.

An Expansion in Infosec

“I have never seen a hacker that required accessibility adjustments. They just don’t exist, so why bother catering for a market when none of them need it?”

This is a quote from someone I argued with once. You see. Here’s the thing.

The reason why you don’t see hackers that require accessibility because hacking has never been accessible.

It’s like saying:

“Why don’t we see more students applying to our internships?”

And then on the application page for those internships:

- 10 years WebDev experience required

It’s literally an oxymoron.

Can't apply to my dream job on Linkedin because "Minimum 12 years' experience in Kubernetes" is needed. #Kubernetes is <6 years old.

@brendandburns @jbeda @cmcluck @tiangolo pic.twitter.com/wDbqfsy3Ga

— Ashish (@ashishwt) July 16, 2020

SEO

Accessibility isn’t just screenreaders. It’s a whole host of things.

Most hacking platforms I’ve come across just aren’t mobile-friendly. At all. This is very weird to see. Any large platform or medium-sized blogger will tell you that

SEO is the reigning champion of views, clicks and good analytics.

Good SEO is a godsend, something that is impossible to achieve unless you work for it.

Google uses a “mobile-first indexing” system. That means that when Google checks your website out to see how to rank your site in their search results (SEO), they predominantly use the mobile browser.

If your website is not mobile-friendly, you will have extremely bad SEO.

Mobile friendly is an accessibility issue, as not everyone can afford laptops or desktops. Mobile friendly is the absolute bare minimum for SEO of a website since.... 2012?

I don't need to explain why SEO is important. Either you understand why it's important, or you're not big enough yet to understand why SEO is important. Either way, it's important.

Better Experience for Everyone

Accessibility improves the experience for everyone. As an example, think of a mobile-friendly site.

TryHackMe has a streaks system. If you go out with your friends and you're about to lose your streak, you could log into the mobile-friendly TryHackMe, complete a question and save your streak.

The mobile-friendly part is to appeal to people who do not have computers, but it improves the experience for people who do.

All the time we see things changed because of accessibility reasons, and the people that never needed the change now appreciate and enjoy the change more than before.

Hacking Tools

Let's start with the basics; hacking tools. If hacking tools aren't accessible, no matter how hard learning platforms try they will never be accessible.

Let's take a look at what I think are the worst offenders, and what we can do to improve upon them.

Metasploit

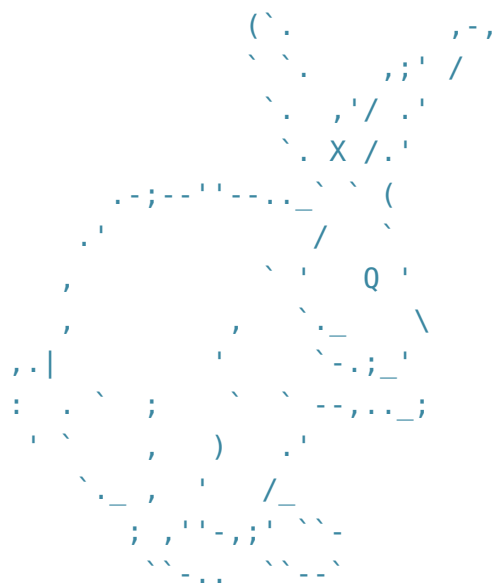
Metasploit is one of the most essential pentesting tools around. And it's entirely text-based, so surely it is accessible? Let's start at Metasploit.

```
Call trans opt: received. 2-19-98 13:24:18 REC:Loc
```

```
Trace program: running
```

```
    wake up, Neo...  
    the matrix has you  
    follow the white rabbit.
```

```
    knock, knock, Neo.
```



<https://metasploit.com>

```

      =[ metasploit v5.0.88-dev ]
+ -- --=[ 2014 exploits - 1097 auxiliary - 343 post ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

```

Metasploit tip: You can upgrade a shell to a Meterpreter session on many pla

```
msf5 >
```

That's a lot of ASCII art for a text-based program. If you put this into <https://www.naturalreaders.com/>, an online TTS (I shorten screenreader to TTS. Note: not all TTS software is a screenreader) program you'd notice it would be okay with the Matrix quote, but then we get the bunny. And let me tell you. That bunny looks nice, but when read back to you it is not fun at all.

Look at how they display information.

```
+ -- --=[ 2014 exploits - 1097 auxiliary - 343 post ]
```

This is also horrific in a screenreader.

Now, enough dunking on Metasploit. They support a `-quiet` mode which turns off the banner and makes it okay.

The only issues that haven't been fixed are typical ASCII art. Such as the `search` command.

Matching Modules

```
=====
```

#	Name	Disclosur
-	----	-----
0	auxiliary/admin/smb/ms17_010_command	2017-03-1

As you might be able to tell, this isn't read very well in a screenreader either.

The quiet mode is good, but it's not good enough.

Sherlock

Sherlock is a rather nice tool for OSINT. But let's take a look at their interface.

```
sherlock johnhammond
```

```
[*] Checking username johnhammond on:  
[-] ResearchGate: Illegal Username Format For This Site!  
[-] 2Dimensions: Not Found!  
[-] 3dnews: Not Found!  
[-] 4pda: Not Found!  
[-] 500px: Not Found!  
[-] 7Cups: Not Found!  
[-] 9GAG: Not Found!  
[+] About.me: https://about.me/johnhammond
```

The `[*]` is nice for people who can see them, as it visually breaks things apart.

However, Sherlock does not have any options to get rid of this.

In TTS software, listen to how it pronounces this:

```
[*] Checking username johnhammond on:
```

It's not very nice. Not to mention that Sherlock prints everything its tried, even if it has failed — with no argument to only print the successful things.

The Bottom Line

Normally GUIs are the ones with A11Y (accessibility, 11 letters between A and Y) problems, but in the hacking subculture, it is clear many hacking tools also have A11Y problems.

They are text tools, but their text causes them to not be accessible.

The absolute bare minimum should be an argument, a flag which enables an accessible mode that gets rid of the unnecessary ASCII and just keeps the important parts, as text, as they should be.

Don't forget that a lot of text sucks too. As sighted people, we can easily see the green text with `[+]` and only read the successes. Each failure of Sherlock is printed to the screen, and there is no way to turn this off.

So every single person using TTS will have to go through the 200+ social media sites Sherlock has, even if they do not return successful?

What You Can Do as a Tool Creator

- Run your terminal app through a screenreader. Use Google Translate if you want.

Try and see if you can get through the entire output of your app from start to finish. If you find it too boring, if it's reading out ASCII titles like "———" or if it takes 20 minutes to read out your basic app — change them.

Google Translate isn't the most advanced TTS system in the world, but it is the absolute bare minimum you can do.

- Ensure that your app doesn't use an extremely weird colour scheme
- Ensure there isn't a bunch of text printed out all on one line. Some people, myself included, find it incredibly hard to read very long text with no paragraphing.

So long as you use common sense and imagine what its like for people of all backgrounds to use your tool, it is better than nothing and will be appreciated.

Box Creators

This section is dedicated to everyone who makes rooms, boxes, or whatever you call them.

Now, let's assume the tools we're using are made to be accessible. How will we go around making sure rooms are accessible?

This section covers both walkthroughs and actual VM-based boxes. TryHackMe uses walkthrough rooms to teach a concept (imagine a blog post), and challenge rooms as your typical CTF boxes.

Pictures

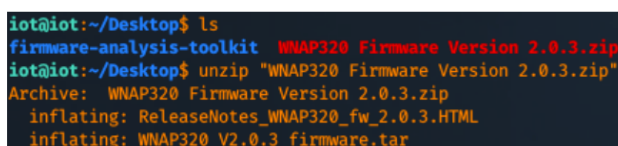
Pictures are easily the most inaccessible part of a room. And easily my largest pet peeve when it comes to these things. Let's take a look at both images in walkthroughs and images in CTFs.

Walkthroughs

Firstly, I'm not sure how this has happened — but don't use images to explain core concepts, lines of code, or anything that needs to be copied and pasted.

Let's look at a real-life example.

Let's unzip the archive.

A terminal window screenshot with a dark background and light-colored text. The prompt is 'iot@iot:~/Desktop\$'. The first command is 'ls', which lists 'firmware-analysis-toolkit' and 'WNAP320 Firmware Version 2.0.3.zip'. The second command is 'unzip "WNAP320 Firmware Version 2.0.3.zip"', which shows progress for inflating 'ReleaseNotes_WNAP320_fw_2.0.3.HTML' and 'WNAP320_V2.0.3_firmware.tar'.

```
iot@iot:~/Desktop$ ls
firmware-analysis-toolkit  WNAP320 Firmware Version 2.0.3.zip
iot@iot:~/Desktop$ unzip "WNAP320 Firmware Version 2.0.3.zip"
Archive: WNAP320 Firmware Version 2.0.3.zip
  inflating: ReleaseNotes_WNAP320_fw_2.0.3.HTML
  inflating: WNAP320_V2.0.3_firmware.tar
```

As you can see, it dropped the release notes and another TAR archive. Let's extract that one too.

Sorry, Cmnnatic.

In this screenshot, we see the text:

Let's unzip the archive.

And then a screenshot of a terminal window, where the creator unzips the archive using a Linux command.

Firstly, accessibility aside, this is a stupid idea.

No one can copy and paste the command from an image. This command looks rather complicated for a complete newbie. You cannot just copy and paste from an image. So why use one?

Not to mention the fact that the image doesn't scale properly. Try looking at this on a mobile device. Have fun! Or perhaps you need to use a much larger font to see the text. You cannot increase the font size of a screenshot.

This is one of the worst ideas I have ever come across. You know earlier how I said making things accessible improves the experience for everyone else? Imagine you didn't have any accessibility requirements. By pushing this to be accessible you will gain:

- Copy and paste the command
- Can easily read it no matter the screen size
- You won't run into many problems when the command can be copied and pasted
- You will have to work out what the command is compared to the output. In the screenshot the `$` is the same colour as the `unzip` command. To a complete newbie they may think you need the "\$" for the command.

So now we understand that screenshots of code or commands are bad for non-accessible reasons, let's talk about the accessible reasons.

If someone were to use a screenreader that just so happened to have OCR support (which, by the way, I have never heard of) then guess what? It will read everything in that image.

Copy and paste this into a TTS program.

```
iot@iot:~/Desktop$ ls
```

I can assure you, this is horrific in TTS.

Now you may be thinking "Oh don't worry, it has alt text which has the command".

1. It doesn't have alt text (by the way, Google uses alt text for SEO reasons).
2. Even if it did, you can't copy and paste alt text. Good luck trying.
3. The screen reader will read "an image of 'ls'". The user will then have to manually write out the command

I'm going to recreate this part of the task in an accessible format to show you how much easier it is.

Let's unzip the archive.

```
ls Find the .zip file we want to unzip. In this case for me, it's called "WNAP320 FirmWare Version 2.0.3.zip"
```

```
unzip "WNAP320 FirmWare Version 2.0.3.zip"
```

This text will easily expand to all screen sizes. Mobile, laptops, 4k monitors and more.

You can easily copy & paste the commands.

I explain how to choose the file to unzip. This is a lot less ambiguous than simply showing the output of "ls" and assuming the user will understand.

No need to use alt text or images.

The font size changes depending on the user's personal settings.

Overall, not only is this a better choice for non-accessible reasons, but it is a better choice for accessible reasons.

Do not use images for code.

If you are sad because the website you are using doesn't support syntax highlighting and just leaves you with a box of code, this is not your problem. This is the websites problem. Submit anyway with the "bad" highlighting and let the website work it out.

Stop giving the website the alternative of just using images. Force their hand into adding syntax highlighting. Do not lower yourself to a level where A11Y is ignored, force the website to raise itself to a standard of good A11Y.

Now, let's talk about something different.

Don't Rely on Images to Tell Your Story

Images are intrinsically inaccessible to almost everyone.

They are hard to read, their colour schemes may put people off, colour-blindness is a thing, they don't scale.

Images are just bad mojo.

Don't use images when you want code.

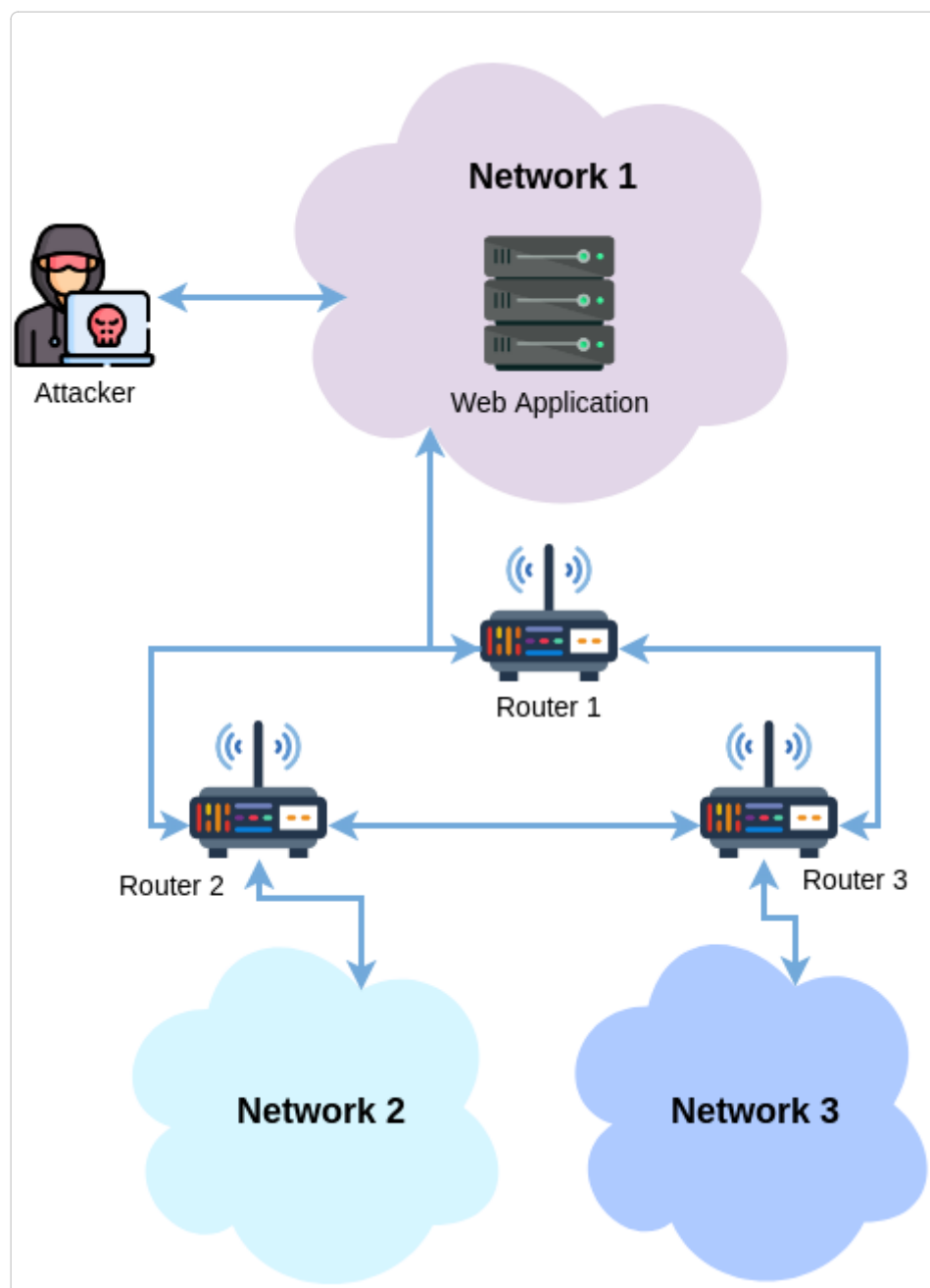
Don't use images when you want quotes.

Don't use images when you want paragraphs of text.

Try to stick closely to the HTML and not to the images.

Now, here's the cool thing.

If you have a network graph, such as this:



And you have to write a lot of alt text to explain that graph.... Maybe you should just, pull the alt text out of the image and label the image like normal text?

This way everyone benefits from a clear explanation.

Alt-text is used for things that visually impaired cannot see, but those who are sighted can. But, we should also be mindful of those who are sighted.

Look at this picture of Ada Lovelace:



If you have a question such as:

“what colour is in her hair”

You will need to explain in alt text what Ada looks like, but not in normal text as sighted people can see that.

Not all images need alt text. Fun fact.



In this task in a walkthrough room, the image on the right-hand side (of someone physically placing blocks of apps into an iPad) does not need alt text.

It does not add to the narrative, it does nothing other than to visually break up the text for the sighted (which, by the way, you should do for A11Y reasons. Not everyone can read entire chunks of text with no images).

When you don't add alt text to an image, most screenreaders will simply skip over the image.

To a visually impaired user, the images add nothing to the story. Imagine alt text. If you are sighted, you probably won't need to read the alt text of every image. It's the same with images. If the image doesn't add to the story, it doesn't need to be read out loud.

For sighted users, we will still want to break up large walls of text.

I am sighted, but I cannot read anything that is 1000 words of text with no breaks in between.

This is basic SEO but also A11Y. Here are some ways you can break up the text:

- Images
- Quotes
- Code blocks
- different sized text
- Italic & bold text

- Unordered Lists
- Ordered Lists

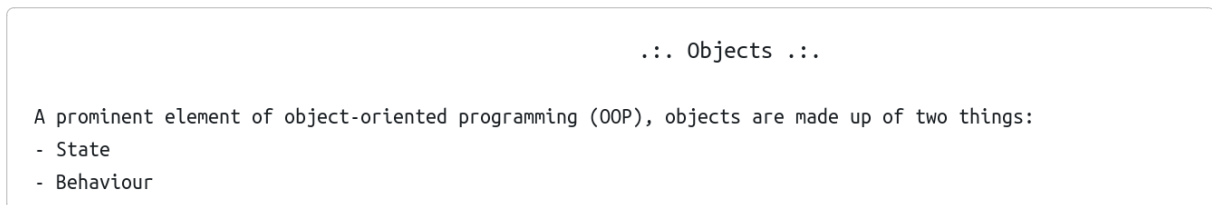
The general order of images should be:

1. No image
2. Image with no alt text (because it is explained in the body, or it is not an image that needs alt text).
3. Alt text.

Fancy ASCII Markup

Do not use ASCII instead of HTML elements like `<h1>` .

Take a look at this:



That header may look fancy, but let's take a look at the HTML.

```
<span style="font-size:18px">... Objects ...</span>
```

So not only is it not a header tag (instead, the font size is just slightly increased) but instead of using a header, it uses “...” to signify it's a header.

Let me tell you, reading this through TTS is not fun. It quite literally reads:

“full stop colon full stop Objects full stop colon full stop”

Now, let's talk about header tags.

Screenreaders are very good at identifying headers. When you read a book, we

often read like this:

Chapter 1. The Unknown Fawn. [1 second break]. Daisy and her....

It's just more natural. Screenreaders know this and can make the reading experience more natural. But completely forgoing basic HTML to create your own header is just bad.

This goes for any ASCII markup. Instead of using a horizontal rule people often use dashes. I don't understand how it's 2020 and people still forgo the use of these basic HTML tags to create their own Frankenstein tags or to try and recreate basic HTML using ASCII only.

It's just bad practice and takes more work than using regular HTML.

The worst part is that the editor for writing this has buttons you can click which use HTML tags. But instead, people forgo clicking a single button to make their own tags, or to use some weird ASCII markup to "look cool". You are not cool for deliberately excluding a vast majority of people from using your work.

The Unknown Fawn

Now, there's 2 things about this.

1. This is completely unreadable in TTS.
2. This won't scale well on any devices other than the creators device.

I don't understand how it's 2020 and people still forgo the use of these basic HTML tags to create their own Frankenstein tags or to try and recreate basic HTML using ASCII only.

It's just bad practice and takes more work than using regular HTML.

Testing your Box

There are 2 tools you can use to test your room which are better than nothing.

1. Google Lighthouse

Google Lighthouse will rank your website (or room, in this case) on a scale of 0 to 100 on a whole bunch of metrics, and it will tell you how to improve upon them.

Most importantly, Google Lighthouse will rank your website (or blog post/walkthrough) for accessibility and will tell you how to improve upon it.

It's not the best tool in the world, but it is a start.

2. Any online TTS

Grab your text and paste it into a TTS. Listen to it back playback to you.

Conclusion

While I mainly talked about TTS here, there is so much more to A11Y than visually impaired. Check out Mozilla's docs on A11Y for more info.

CTFs

Now let's talk about CTFs. Traditionally when people think of A11Y Hacking they think of walkthroughs. But CTFs can also be A11Y.

Due to the nature of operating systems, most OS' are A11Y friendly (especially things such as Ubuntu). And especially if you only access it through a terminal interface.

Unlike walkthroughs, there isn't much to do to make a CTF accessible. Although I'll admit, I've never tried this — but I can guess what the biggest hurdles will be.

- The tools that are used

Forcing the user to use something like Metasploit or Sherlock, which are inherently inaccessible (until they change) will make your CTF inaccessible.

For Metasploit it's quite easy to point to a manual exploit and say "try this instead", but be wary of the tools you use. At the start of the article, I showed you how to make accessible tooling.

Use that information to decide on the tools required for your CTF.

- Websites, documents, or other items

A lot of the time in a CTF you'll present the user with a website. Be mindful if you are creating a custom-built website, make sure it adheres to A11Y.

Otherwise, make sure the website isn't horrific for A11Y purposes.

There was this one CTF where one of the flags was an image on the website. Try to avoid hiding flags in images if you can.

Not only is steganography not fun, but it's also inaccessible. I'm not saying not to do it. If you have an amazing stego idea, go ahead. But try to avoid it if you can. Stego is a cool skill to learn, but it's not for everyone.

There isn't much more to talk about that hasn't been covered already, but I will mention one thing.

The Try Harder Mentality

Imagine this.

You are competing in a CTF on HackTheBox. One part of this CTF is that you have to listen to a sound file which contains some kind of encrypted text.

You ask for help, and immediately you get bashed with "Try Harder KEKW".

Try Harder is an alright mentality to push yourself with, but when it is physically impossible for the person to try due to the box creators incapacity to consider A11Y try harder is toxic.

Try Harder mentality doesn't work when the room is inaccessible, and it especially

doesn't work when every room is inaccessible and you can't try at all because of it.

But you shouldn't just disclude people from entire challenges because it is inaccessible to them. All it takes is someone to say "hey can anyone help me, I can't hear the audio due to a disability" and for a mentor to reply with "sure, let me DM you the transcript :)" to help make the room, the mood, and the community a more wholesome place.

It's all about Try Harder until it is physically impossible for you to try harder, or for you to try at all.

Infographics



Fancy ASCII art is not accessible.

3 USE REAL HTML ELEMENTS

Please do not make custom HTML elements. Either edit the current ones with CSS (if possible), or use the default settings. Don't increase font size thinking its equivalent to `<h1>`.

4 CLEAR WRITING

Explain the why with clear intentions.

5 BE WARY OF NON ACCESSIBLE TOOLS

Do you require tool X for your room, but that tool is not accessible? Your room is not accessible.

6 MAKE YOUR WEBSITES A11Y

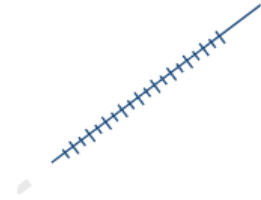
If you build a custom website for your CTF, please just make it maintain basic A11Y. Same principles as here, but applied to a website.

7 TRY HARDER? NO.

If there is something in the room that cannot be solved due to a disability, please let people know so we can help people who can't try, yet alone try harder.

8 TOOLS TO CHECK

- Google Lighthouse
- Any online TTS program



CTF writeups are literally just blog posts read any A11Y blog about making them better, like this one.

Website A11Y

I've already discussed hacking tools and CTFs, but let's talk about websites. This is for you, developers of TryHackMe, HackTheBox, ImmersiveLabs & more.

If the room is accessible, and the tools are accessible, but your website is not — you are the problem.

As an organisation, you can hire someone who is an expert in A11Y, or you can promote an employee to study A11Y. Either way, I'm not qualified to teach you. So instead, I will show you why.

Take a look at this exemplary person, Elvis. Elvis learnt to code on their phone, created an app (from their phone) and now works for an MIT startup.

This happened because the websites used to learn to code or code themselves were mobile-friendly.

Elvis changed his life because privileged people chose to make their sites accessible.

Imagine this. Imagine being told one day that your website changed someone's life like this.

The vast majority of the world do not own laptops or desktops. They use mobile phones. To make the website non-accessible on a mobile you are missing a very large market.

Think of how many hacking websites are mobile related. I'll tell you something. I have never seen a single one.

From a business perspective, this is a unique selling point. Make the worlds first mobile-friendly hacking website and destroy your competition, get millions and

millions of new users from all over the world who can only access the web via their phones.

Get sight-impaired users, hard of hearing users.

Be accessible, be kind, and you will thrive.

From a business perspective, you want good SEO? You need accessibility.

Google ranks websites based on their mobile performance, as well as ease of readability, use of alt text to describe images and much more.

Google is accessible first by its very nature.

If you want great SEO, you need to make your website accessible.

I'm not an SEO expert either. But consult an SEO expert and they will tell you accessibility is #1.

Heck, if you hire or train an A11Y dev you likely won't need an SEO dev. Your website is unique, unlike blogs or recipe sites. By the very nature of the category of the website you own, all you need to do is focus on accessibility and gaining more users and you'll increase your SEO substantially.

After this, if you want a slight increase in SEO to consult the experts. But if you do that now, I can almost guarantee they'll just say "accessibility is important".

Conclusion

This is it for my rant on making hacking accessible.

I'm a moderator for TryHackMe, so if you ever need any help or have qualms with this article message me via TryHackMe :)

Infosec

A11Y



© 2022 Skerritt.blog · Powered by Hugo · Theme PaperMod