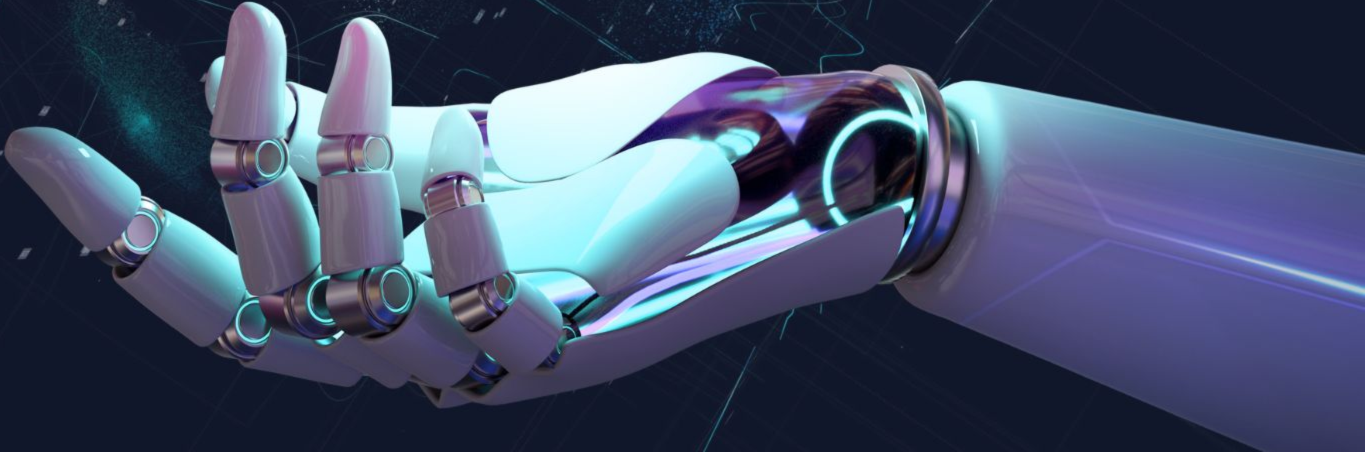
I’m Not A Robot

CAPTCHA DEVELOPMENT & SECURITY ANALYSIS

Asmi, Kelly, Kenan, Jon, Bianca, Lucy



High Level Questions

1. **Vulnerability Mapping**: What are the most common methods used to bypass CAPTCHA, and how effective are they against modern CAPTCHA versions?
2. **AI Resilience:** How well do CAPTCHAs hold up against deep learning models?
3. **Cost vs. Security**: What trade-offs exist between maintaining high security and the computational resources needed to develop and maintain complex CAPTCHA systems?

Project Topic & Solution

- Focus: Developing & understanding CAPTCHA systems to improve security against automated bot attacks
- Solution Approach:
  - Literature Review: Analyzed evolution from text-based to complex image-recognition CAPTCHAs.
  - Vulnerability Analysis: Tested slider-based and text-based CAPTCHAs, revealing weaknesses like predictability and lack of server-side validation.
  - Proposed Enhancements:
    - **Dynamic elements**: Introduce randomized starting conditions and server-side checks.
    - **Balanced Design**: Enhance security without compromising accessibility, especially for differently-abled users.

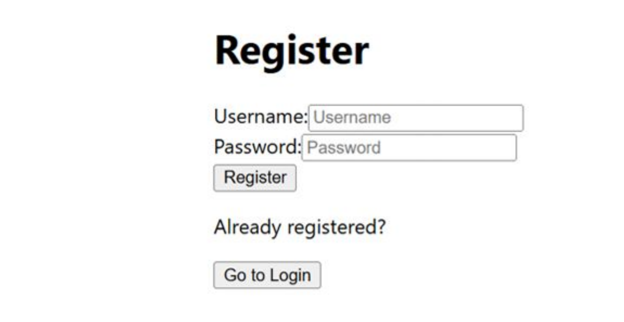
Timeline & Deliverables

- Week 4: Preliminary Literary Review
- Week 5: Functional Login Page with CAPTCHA’s
    - Currently focusing on Image Based (puzzle, rotated image, select relevant image etc)
- Week 5-6: Attack related-literature review
- Week 6-7: Attacks Implemented on ^
- Week 7-8: Defenses Implemented / Suggested
- Week 9: Report Completed

Threat Model

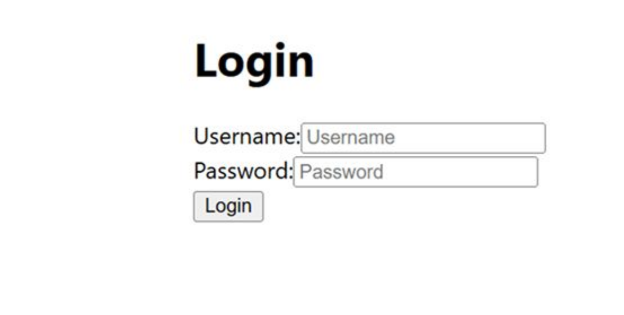
- Attackers: Malicious developers, cybercriminals, competitors
- Attack Vector: CAPTCHA Component on webpage (use automated scripts with web scraping/OCR techniques etc)
- Assets at Risk: Protected resources, sensitive user data (financial records, educational records etc)
- Attack Steps:
  - Prepare script to navigate target page and capture CAPTCHA.
  - Use OCR for text extraction from CAPTCHA images.
  - Automate CAPTCHA response submission to access protected areas.
- Assumptions: White-box, visual-based CAPTCHA, no server-side validation

Our (Basic) Setup



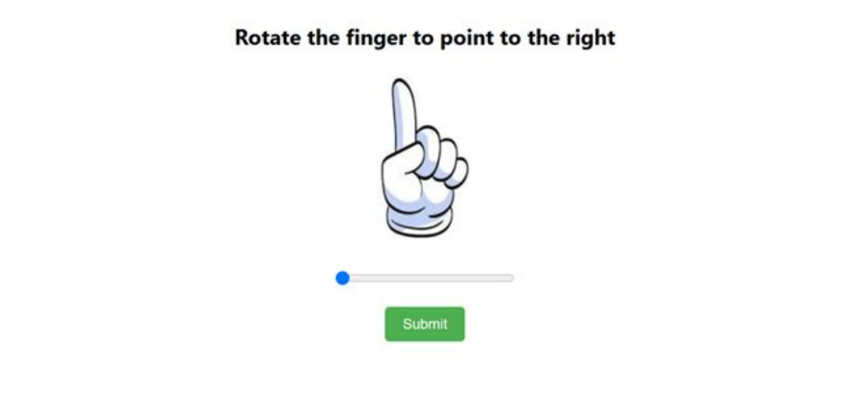
Register

FIGURE 1. **Screenshot of login page**



Login

FIGURE 2. **Screenshot of registration page**



Rotate the finger to point to the right

FIGURE 3. **Image Based CAPTCHA: Rotate to match specified direction**

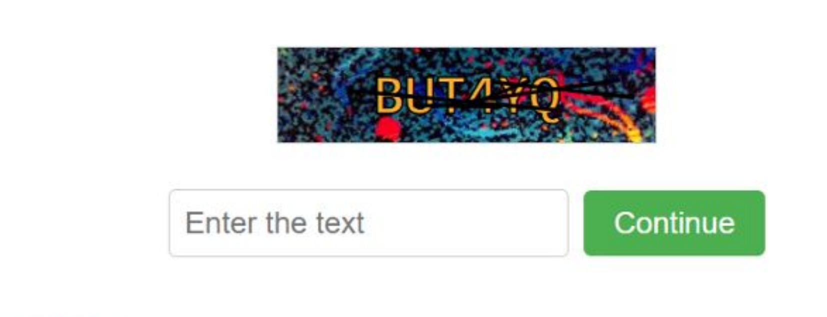
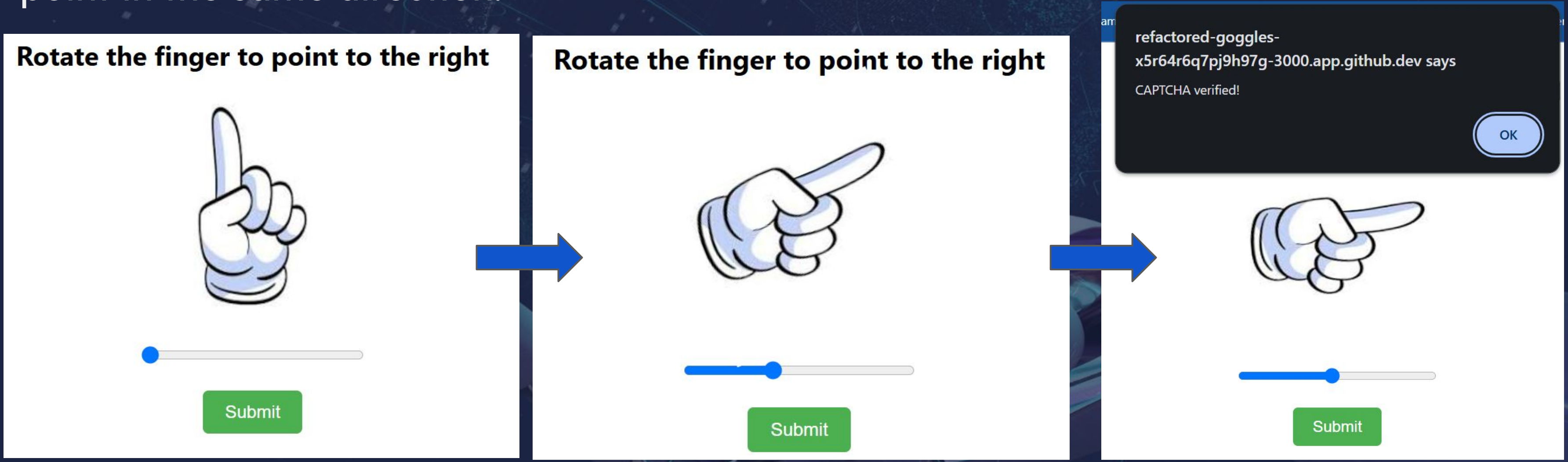


FIGURE 4. **Text Based CAPTCHA: Type in a matching string**

Captcha #1: Rotate an Image

Implemented a basic image based Captcha.

Major vulnerabilities: Image always starts in the same orientation, always asks to point in the same direction.



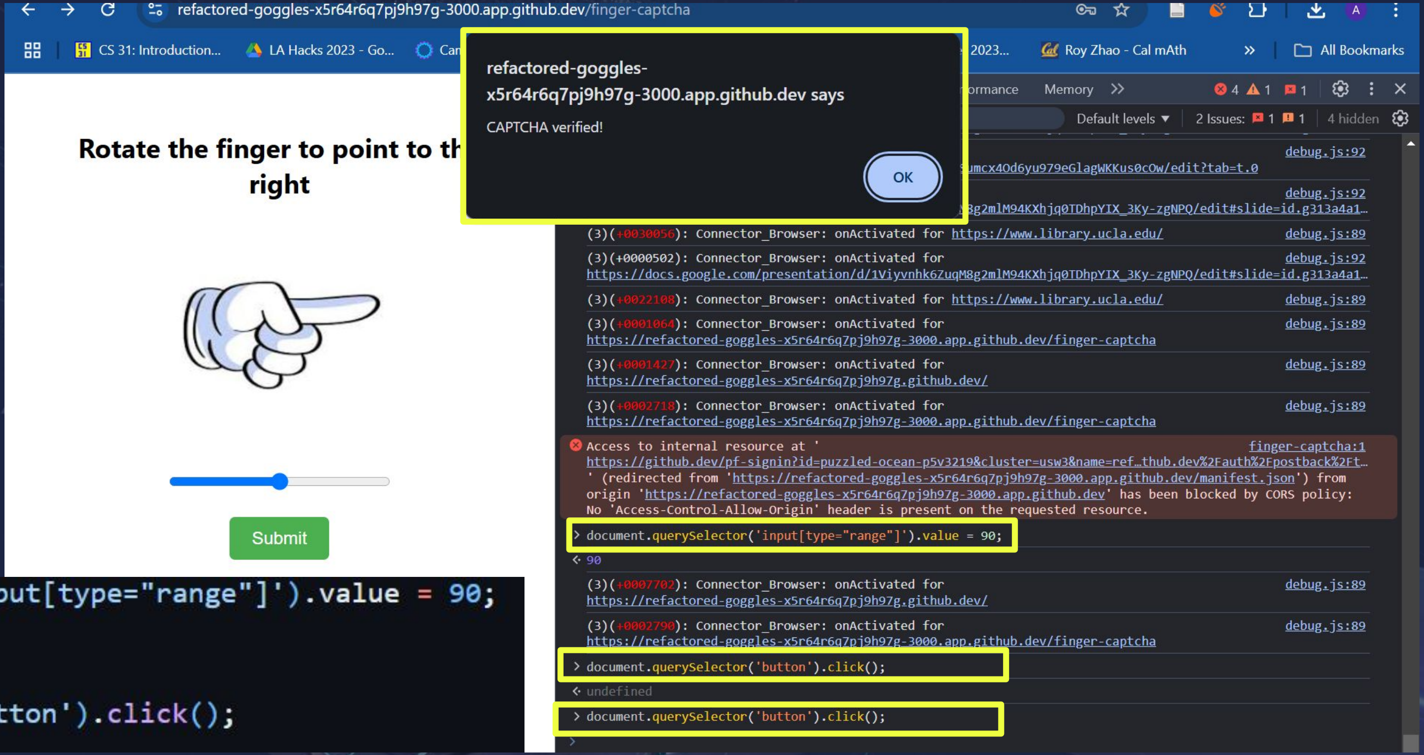
Rotate the finger to point to the right

Rotate the finger to point to the right

refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev says

CAPTCHA verified!

Attack #1: Rotate an Image



Rotate the finger to point to th right

refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev says

CAPTCHA verified!

```
                                                                    debug.js:92
umcx4Od6yu979eGlagWKKus0cOw/edit?tab=t.0
                                                                    debug.js:92
8g2mlM94KXhjq0TDhpYIX_3Ky-zgNPQ/edit#slide=id.g313a4a1…
(3)(+0030056): Connector_Browser: onActivated for https://www.library.ucla.edu/   debug.js:89
(3)(+0000502): Connector_Browser: onActivated for                   debug.js:92
https://docs.google.com/presentation/d/1Viyvnhk6ZuqM8g2mlM94KXhjq0TDhpYIX_3Ky-zgNPQ/edit#slide=id.g313a4a1…
(3)(+0022108): Connector_Browser: onActivated for https://www.library.ucla.edu/   debug.js:89
(3)(+0001064): Connector_Browser: onActivated for                   debug.js:89
https://refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev/finger-captcha
(3)(+0001427): Connector_Browser: onActivated for                   debug.js:89
https://refactored-goggles-x5r64r6q7pj9h97g.github.dev/
(3)(+0002718): Connector_Browser: onActivated for                   debug.js:89
https://refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev/finger-captcha
Access to internal resource at '                                    finger-captcha:1
https://github.dev/pf-signin?id=puzzled-ocean-p5v3219&cluster=usw3&name=ref…thub.dev%2Fauth%2Fpostback%2Ft…
' (redirected from 'https://refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev/manifest.json') from
origin 'https://refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev' has been blocked by CORS policy:
No 'Access-Control-Allow-Origin' header is present on the requested resource.
> document.querySelector('input[type="range"]').value = 90;
<· 90
(3)(+0007702): Connector_Browser: onActivated for                   debug.js:89
https://refactored-goggles-x5r64r6q7pj9h97g.github.dev/
(3)(+0002790): Connector_Browser: onActivated for                   debug.js:89
https://refactored-goggles-x5r64r6q7pj9h97g-3000.app.github.dev/finger-captcha
> document.querySelector('button').click();
<· undefined
> document.querySelector('button').click();
>
```

```
document.querySelector('input[type="range"]').value = 90;

//Execute this line twice
document.querySelector('button').click();
```

Defense #1

Main Vulnerability: Static Captcha, predictable start and end state

Suggested Defenses:

- Randomize start/end state of image
- Randomize start position/range of slider
- Adding a timeout period after each set of (e.g 3) failed login attempts to prevent brute force attacks.

Captcha #2: Text Based

Implemented a basic text based CAPTCHA

Major Vulnerabilities/Features:

- Predictable length: 6 characters
- Static background image
- Some variation of lines to obfuscate text
- Lines move each time a char is typed

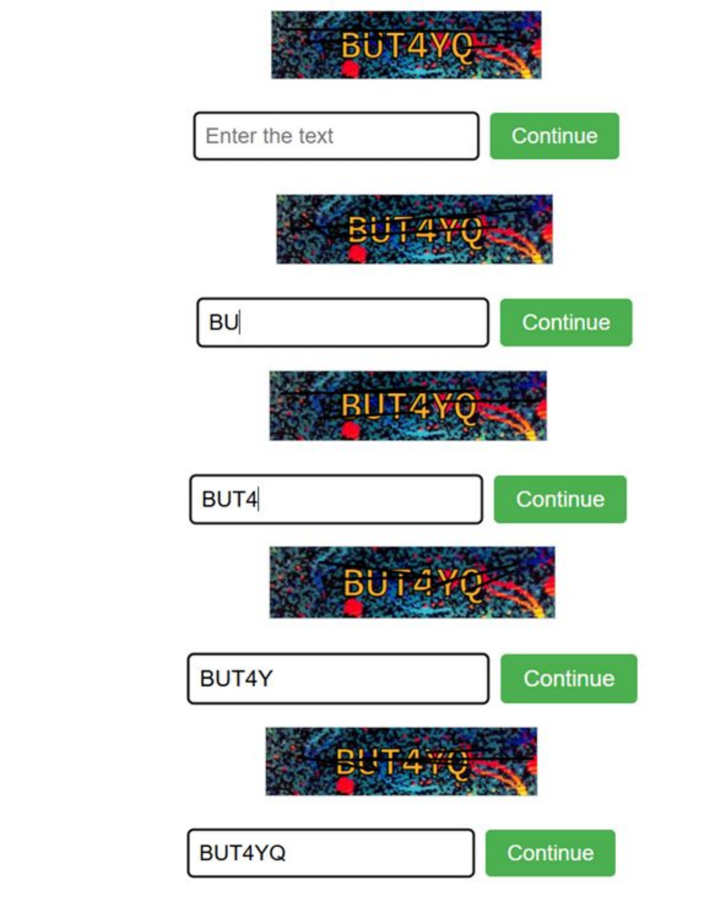


FIGURE 7. Randomized line overlay with every character typed

Attack #2 (In Progress)

The goal is to implement at least one of the following:

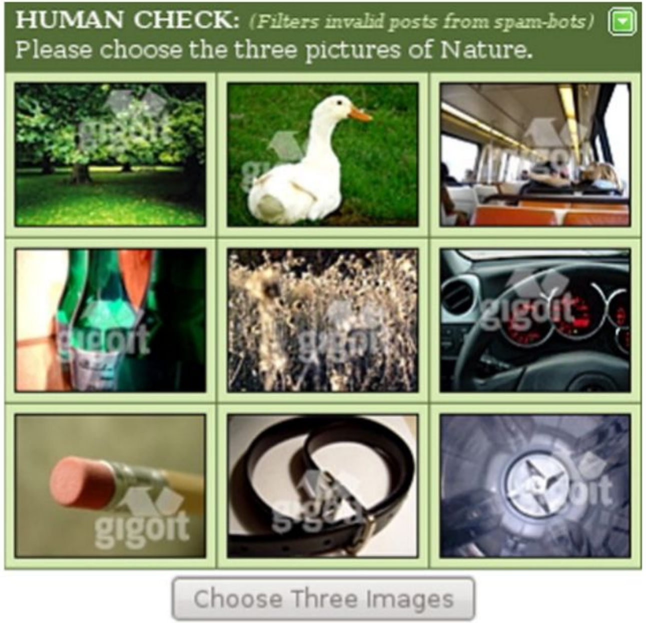
- Brute Force Attack: Always has length 6 and chooses from 36 chars (capital letters and numbers) - (26+10)^6 = 2176782336 possibilities
- OCR Attack: Use a tool like Tesseract.js to analyze CAPTCHA image, automatically reading visible characters
- Exploiting Lack of Server Side Validation: Try to bypass CAPTCHA entirely, manipulating requests directly through HTTP tools.

Defense #2

Suggested defenses (based on the targeted vulnerabilities):

- Increase CAPTCHA Complexity: Introduce more complex background noise, wavy/rotated text and random distortions to mitigate OCR attacks
- Time-Based Expiry: Prevent attacker from capturing valid CAPTCHA tokens and using them in a future attack.
- CAPTCHA Refresh: For each attempt, refresh the CAPTCHA after a specific time interval, ensuring attackers cannot continuously solve it without user interaction.

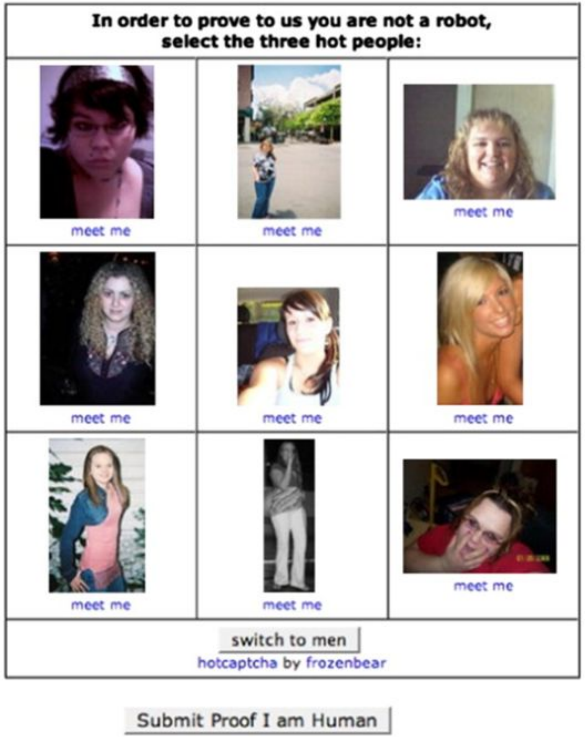
Real World: Captcha Types



HUMAN CHECK: *(Filters invalid posts from spam-bots)*

Please choose the three pictures of Nature.

FIGURE 10. Screenshot of the HumanAuth CAPTCHA



In order to prove to us you are not a robot, select the three hot people:

FIGURE 9. Screenshot of what HotCaptcha looked like.

Real World: CAPTCHA Challenges

- CAPTCHAs can create accessibility issues. A 2024 study argues that rate limiting could be effective in improving security without compromising user experience.

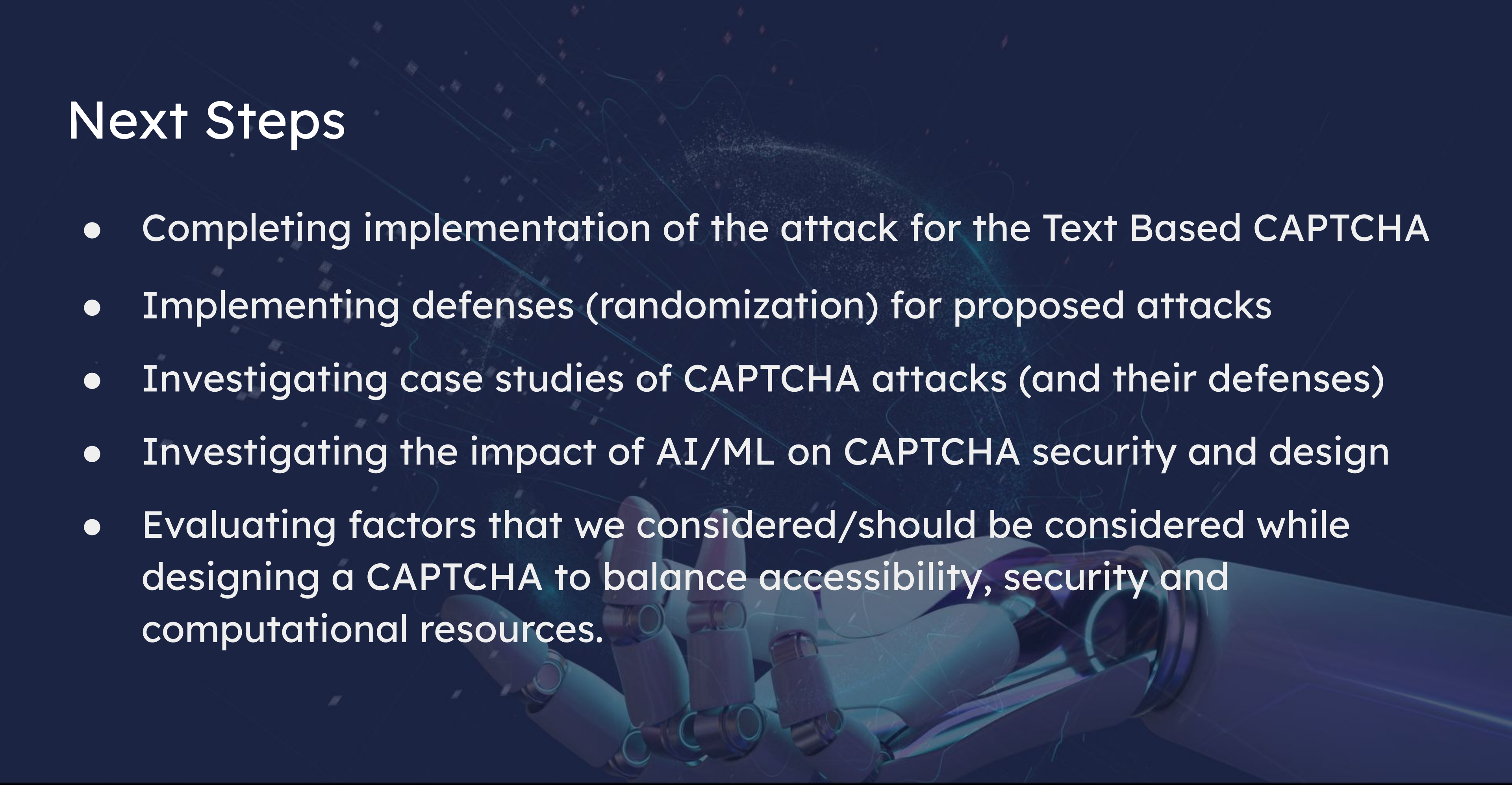
Limitations/Assumptions

Assumptions

- White Box Attack: Attacker is aware of the internal construction of the CAPTCHA and can craft attacks accordingly
- Accessibility: End user does not have visual impairments (the deployed CAPTCHAs in this project rely on visual cues)

Limitations

- Lack of Advanced Obfuscation
- No server-side CAPTCHA validation
- No logging/tracking failed attempts



Next Steps

- Completing implementation of the attack for the Text Based CAPTCHA
- Implementing defenses (randomization) for proposed attacks
- Investigating case studies of CAPTCHA attacks (and their defenses)
- Investigating the impact of AI/ML on CAPTCHA security and design
- Evaluating factors that we considered/should be considered while designing a CAPTCHA to balance accessibility, security and computational resources.