

Overview


This web app is a **collection of 8 beginner-friendly cybersecurity tools** built with **only HTML, CSS, and JavaScript**, designed to simulate real-world cybersecurity concepts and practices in a **user-friendly interface**. It's ideal for students, educators, and enthusiasts looking to understand security principles through interactive mini-projects.


Tools Used

- **HTML5** – structure and content
 - **CSS3** – styling and layout
 - **JavaScript (ES6)** – functionality and interactivity
-

Project List and Cybersecurity Relevance

1. Password Strength Checker

 **Purpose:** Teach users how password complexity affects security.

 **Cybersecurity Concept:** Weak passwords are easy to guess or crack. Strong passwords reduce brute-force attack success rates.

 **How it works:**


- Checks length (8+ characters), uppercase letters, numbers, and symbols.
- Gives a strength rating based on these factors.


 **Code Highlights:**

```
if (val.length >= 8) score++;  
if (/[A-Z]/.test(val)) score++;  
if (/[0-9]/.test(val)) score++;
```

```
if (/^[A-Za-z0-9]/.test(val)) score++;
```

2. Phishing Email Detector

 **Purpose:** Help identify common phishing keywords and suspicious URLs.

 **Cybersecurity Concept:** Phishing is a major attack vector where attackers trick users into revealing sensitive info.

 **How it works:**


- Analyzes user-pasted email content.
- Flags words like "urgent", "verify account", or links (<http://...>) as suspicious.

 **Code Highlights:**

```
const suspicious = ["urgent", "click here", "login now", "verify  
account"];  
if (found.length > 0 || hasLink) // show phishing warning
```

3. File Encryption Tool (XOR)

 **Purpose:** Demonstrate simple encryption logic using XOR.

 **Cybersecurity Concept:** Encryption protects data confidentiality. XOR encryption is a basic example of symmetric key encryption.


 **How it works:**


- User inputs text and a numeric key.
- Each character is XORed with the key.
- Encrypted output is shown in Base64.

 **Code Highlights:**

```
result += String.fromCharCode(text.charCodeAt(i) ^ key);  
btoa(result) // convert to Base64
```

4. Firewall Rule Simulator

 **Purpose:** Simulate basic firewall rule logic.

 **Cybersecurity Concept:** Firewalls block or allow network traffic based on IP rules to protect internal systems.

 **How it works:**


- User inputs an IP address.
- Checks against a hardcoded list of blocked IPs.
- Displays if the IP is allowed or blocked.

 **Code Highlights:**

```
const blockedIPs = ["192.168.1.1", "10.0.0.5"];  
if (blockedIPs.includes(ip)) // blocked
```

5. Packet Sniffer (Simulated)

 **Purpose:** Emulate packet capture output for educational purposes.

 **Cybersecurity Concept:** Packet sniffing monitors data flow across networks. Used by attackers or defenders for analysis.

 **How it works:**

- Generates sample network packets (e.g., DNS, HTTP requests).
- Displays them in real time like a sniffer log.

 **Code Highlights:**

```
const packets = [  
  "192.168.1.10 → 172.217.0.142 [HTTP] GET /index.html",  
  "192.168.1.10 → 8.8.8.8 [DNS] Query A google.com"  
];
```

6. 🧠 Keylogger Demo

📌 **Purpose:** Simulate how keyloggers capture input.

🧠 **Cybersecurity Concept:** Keyloggers are malware that record keystrokes, often used for credential theft.

✅ **How it works:**

- Every keystroke in the input field is logged and displayed.
- Raises awareness about malicious monitoring of user input.

📄 **Code Highlights:**

```
logger?.addEventListener('keydown', (e) => {  
  logged.innerText += e.key + " ";  
});
```

7. 🖼️ Steganography (Simulated)

📌 **Purpose:** Introduce the concept of hiding messages inside files/images.

🧠 **Cybersecurity Concept:** Steganography is the technique of hiding data in plain sight, e.g., embedding in images.


✅ **How it works:**


- User types a secret message.
- Message is Base64 encoded and simulated as embedded in an image.

Code Highlights:

```
const data = btoa(msg); // Base64 simulates hiding  
img.alt = msg; // simulated storage in image metadata
```

8. 📱 2FA TOTP Simulator

 **Purpose:** Mimic Time-based One-Time Passwords (TOTP).

 **Cybersecurity Concept:** TOTP is used in 2-Factor Authentication to add a layer of security.

 **How it works:**

- Generates a pseudo-random 6-digit code every 30 seconds.
- Simulates time-based nature of real TOTP (like Google Authenticator).

Code Highlights:

```
const time = Math.floor(Date.now() / 30000);  
const code = ("000000" + (time % 1000000)).slice(-6);
```

UI Design Principles Applied

Feature	Description
Dark Mode	Modern look, reduces eye strain
Card Layout	Separates functionality for clarity
Responsive Grid	Adapts to mobile, tablet, and desktop
Visual Feedback	Colors and effects on input, hover, and output
Live Preview	Results update instantly after actions



How to Use / Run

1. Copy the HTML code into a file named `index.html`.
 2. Open the file in a browser (Chrome, Firefox, etc.).
 3. Test each tool in the cards.
 4. Optionally, host it on GitHub Pages or Netlify for online access.
-



Learning Outcomes

By completing and using this app, learners will understand:

- The basics of **password security**
 - **Phishing red flags** in emails
 - **Symmetric encryption logic**
 - Simple **network rule enforcement**
 - **Packet structure** and sniffing concepts
 - Dangers of **keylogging**
 - Basics of **steganography**
 - How **2FA/TOTP** enhances account security
-



Disclaimer

These tools are for **educational and demonstration purposes only**. They simulate concepts to help users understand and practice cybersecurity fundamentals in a safe environment.

Github URL: <https://github.com/brahmaputraS/cyber-security-projects.git>

Demo Link: <https://reactsamples-f5783.web.app/>