

IoT Shield

Website Documentation

HTML • CSS • JavaScript • Static Site
Capstone II | 2026

Entry Point	index.html
Styles	css/style.css (2,026 lines)
Scripts	js/banner.js, attacks-slideshow.js, faq.js, contact-modal.js, scroll-top.js, section-nav.js
Assets	assets/images/ (25 images: PNGs and JPEGs)
Fonts	Outfit + DM Sans via Google Fonts
Dependencies	None (no npm, no build step, no framework)
Local Preview	python -m http.server 8080 or .\serve-local.ps1 (port 8765)

Table of Contents

Table of Contents 2

1. Overview..... 4

 1.1 File Structure 4

 1.2 Running Locally..... 4

 1.3 Deployment 4

2. HTML Structure — index.html 6

 2.1 <head> Setup..... 6

 2.2 Page Sections 6

 2.3 Accessibility Markup 7

 2.4 Carousel Markup Pattern..... 7

 2.5 Scripts Load Order 7

3. Styles — css/style.css..... 8

 3.1 Design Tokens (:root) 8

 3.2 Responsive Breakpoints..... 8

 3.3 Key Component Styles 8

 Hero (.hero-panel) 8

 Features Grid (.sf-product-tab-order-list)..... 9

 Slideshow (.attacks-slideshow) 9

 FAQ (.faq-item) 9

 Contact Modal (.contact-modal) 9

4. JavaScript Modules..... 10

 4.1 js/banner.js — Hero Gradient Canvas 10

 Canvas Setup 10

 Gradient Blobs..... 10

 Animation..... 10

 Reduced Motion..... 10

 4.2 js/attacks-slideshow.js — Carousel Controller 10

 Initialization 10

 Navigation..... 11

 Scroll Sync..... 11

 Autoplay..... 11

 Resize Handling..... 11

 4.3 js/faq.js — FAQ Accordion..... 11

 Open / Close Logic 11

 CSS-Only Animation..... 12

4.4 js/contact-modal.js — Contact Dialog	12
Open.....	12
Close	12
Form Submit	12
4.5 js/scroll-top.js — Back-to-Top Button.....	12
Visibility.....	12
Click.....	12
4.6 js/section-nav.js — Sticky Section Navigation.....	13
Hero Departure	13
Viewport Guard.....	13
Reveal on Scroll.....	13
Active Section Tracking	13
5. Images & Assets	14
5.1 Architecture & Diagram Images	14
5.2 Dashboard Screenshots	14
5.3 Attack Example Screenshots (Different Attacks carousel)	14
6. Extending & Maintaining the Site	16
6.1 Adding a Carousel Slide	16
6.2 Adding a FAQ Item	16
6.3 Adding a Feature Card	16
6.4 Wiring the Contact Form.....	16
6.5 Changing Autoplay Speed	17
6.6 Changing the Hero Gradient Colors.....	17
7. Known Limitations.....	18

1. Overview

The IoT Shield website is a single-page static site that serves as the project showcase and public-facing documentation for the IoT Shield capstone. It requires no server-side runtime, no build step, and no package dependencies. The full page is contained in one HTML file, one CSS file, and six JavaScript modules.

The site covers eight content sections, two interactive carousels, an FAQ accordion, and a contact dialog — all progressively enhanced with vanilla JavaScript and fully accessible.

1.1 File Structure

File / Folder	Purpose
<code>index.html</code>	Single-page site markup — all sections, carousels, FAQ, contact dialog, footer
<code>css/style.css</code>	All styles: design tokens, layout, components, responsive rules (2,026 lines)
<code>js/banner.js</code>	Hero section animated gradient canvas
<code>js/attacks-slideshow.js</code>	Carousel controller — shared by both slideshows
<code>js/faq.js</code>	FAQ accordion — single-open behavior
<code>js/contact-modal.js</code>	Contact dialog — open, close, submit
<code>js/scroll-top.js</code>	Back-to-top button visibility and scroll
<code>js/section-nav.js</code>	Sticky dot navigation rail (wide screens only)
<code>assets/images/</code>	25 images: dashboard screenshots, attack examples, architecture diagrams
<code>serve-local.ps1</code>	Optional PowerShell script: runs Python server on port 8765

1.2 Running Locally

From the project root, run either of the following:

```
python -m http.server 8080 --bind 127.0.0.1
```

Then open `http://127.0.0.1:8080/` in a browser. Or use the included script:

```
.\serve-local.ps1
```

The script uses port 8765 and opens the browser automatically. Stop with Ctrl+C. If 8080 is occupied, use any free port. Do not open `index.html` directly as a `file://` URL — some browser security policies block local font and image loading.

1.3 Deployment

- Upload `index.html`, `css/`, `js/`, and `assets/` to any static host (GitHub Pages, Netlify, S3 + CloudFront, etc.).
- No server-side runtime or database is required.

- Fonts load from Google Fonts CDN — ensure outbound HTTPS to fonts.googleapis.com and fonts.gstatic.com is not blocked.
- Connect the contact form (#contact-form) to a backend or form service before going live — it is currently a client-side demo only.

2. HTML Structure — index.html

The page uses semantic HTML5 with a single `<main>` element, landmark regions, and ARIA attributes throughout. Scripts are loaded at the end of `<body>` in dependency order. No JavaScript framework or bundler is used.

2.1 `<head>` Setup

- charset UTF-8; viewport with `viewport-fit=cover` for notched devices
- theme-color `#070c24` and color-scheme dark for native browser chrome
- Google Fonts preconnect + stylesheet link for Outfit and DM Sans
- Single stylesheet: `css/style.css`

2.2 Page Sections

ID / Element	Nav Label	Content
<code>#banner-843</code>	IoT Shield	Hero: animated canvas gradient, eyebrow text, h1 with teal accent on 'Shield', pill CTA button linking to <code>#section-platform</code>
<code>#section-platform</code>	(within <code>#cta-384</code>)	Platform intro: cs-topper, h2, two paragraphs describing IoT Shield
<code>#section-problem</code>	The Problem	h3, descriptive paragraph, 5-item problem-list with custom bullets
<code>#section-solution</code>	Our Solution	Flex row: copy on left (h3 + 2 paragraphs), architecture image on right inside <code>.iot-visual-wrap</code> with layered bg gradients
<code>#section-features</code>	Key Features	h2, intro paragraph, 4 <code>tabindex=0</code> cards in <code>.sf-product-tab-order-list</code> : Real-Time Monitoring, Centralized Dashboard, Threat Detection, Custom Alerts
<code>#section-attacks</code>	Different Attacks	12-slide auto-play carousel. Each slide is an <code><article></code> with <code><figure></code> + <code><figcaption></code> . Toolbar: prev/next buttons + auto-generated dot buttons + aria-live status line.
<code>#section-how-it-works</code>	How it works	10-slide platform walkthrough carousel (same markup pattern, <code>data-autoplay-ms=7000</code>). Followed by <code>.how-it-works-explain</code> block with h3 + project summary prose.
<code>#faq-1193</code>	FAQ	Two-column layout: image card on left (aria-hidden), 7-item accordion on right. Native button triggers, aria-expanded, CSS <code>grid-template-rows</code> animation.
<code>#cta-697</code>	Contact	Centered headline + 'Get in touch' button. Inline SVG wave decoration below.
<code><footer></code>	—	Brand name + course/capstone tagline. No links.
<code>#contact-modal</code>	—	Native <code><dialog></code> : close X button, intro copy, form (name/email/phone/referral/message), success state panel.

2.3 Accessibility Markup

- Skip link: `<a class="skip" href="#main-content">` as first element in `<body>`
- Sticky nav: `aria-label="On this page"`, `aria-hidden` toggled by `js/section-nav.js`
- Carousels: `role="region"` `aria-roledescription="carousel"` on viewport; inactive slides get `aria-hidden="true"`; status line is `aria-live="polite"` `aria-atomic="true"`
- FAQ: each trigger button has `aria-expanded` and `aria-controls`; each panel has `role="region"` `aria-labelledby`
- Contact button: `aria-haspopup="dialog"` `aria-controls="contact-modal"`
- Contact dialog: `aria-labelledby` switches between `contact-modal-title` and `contact-success-title` depending on form state
- Images: all `<img>` elements have descriptive alt text; decorative image column in FAQ uses `aria-hidden="true"` on the parent

2.4 Carousel Markup Pattern

Both carousels share identical markup. The root div carries the initialization attributes:

```
<div class="attacks-slideshow" data-iot-slideshow data-autoplay-ms="6500">
```

Inside the root: a viewport div (`role=region`, `aria-roledescription=carousel`, `tabindex=0`), `article.attacks-slide` elements each with `data-label` and a figure/figcaption, a toolbar div with prev/next buttons and a `.attacks-slideshow-dots` container (dots injected by JS), and a status paragraph.

2.5 Scripts Load Order

All six scripts are loaded at the bottom of `<body>` with no `defer` or `async` attributes, ensuring the DOM is ready:

```
js/banner.js → js/attacks-slideshow.js → js/faq.js → js/contact-modal.js → js/scroll-top.js → js/section-nav.js
```

3. Styles — css/style.css

The stylesheet is 2,026 lines of vanilla CSS. It uses custom properties (CSS variables) as a design token system and relies on modern layout features: CSS Grid, Flexbox, scroll-snap, and grid-template-rows animation for the FAQ.

3.1 Design Tokens (:root)

Token	Value	Used For
--primary	#5eead4 (teal)	CTA buttons, accent lines, feature hover glow
--secondary	#7c3aed (violet)	Gradient blobs, feature number color, nav dots active
--surface	dark blue-gray	Section backgrounds
--surface-elevated	slightly lighter	Cards, modal panel, FAQ items
--text-muted	muted gray	Eyebrows, captions, secondary copy
--border-subtle	low-contrast line	Card and panel borders
--radius-sm	6px	Small elements
--radius-md	14px	Cards, panels, image containers
--radius-lg	24px	Modal, large containers

3.2 Responsive Breakpoints

Breakpoint	Changes
1320px	Sticky section nav becomes visible (position: fixed, top-right rail)
900px	.solution-with-media stacks vertically (flex-direction: column); .faq-layout stacks; image containers center
768px	.cs-container horizontal padding reduces via clamp(); FAQ row min-height and font-size tighten; features grid goes single-column
540px	Contact modal switches to bottom-sheet style (rounded top corners, slides up from bottom)
480px	Hero panel, solution image, and FAQ image scale with min() to prevent overflow on small phones

3.3 Key Component Styles

Hero (.hero-panel)

- Frosted-glass panel: backdrop-filter blur + semi-transparent background
- Eyebrow: .hero-eyebrow in small caps with letter-spacing
- .cs-int-title: large display font (Outfit), .title-accent applies teal gradient text via background-clip: text

- `.cs-button-solid`: pill shape, teal fill, flex row with SVG arrow icon
- `is-departing` class: CSS transition that lifts and fades the panel on Learn More click

Features Grid (`.sf-product-tab-order-list`)

- CSS Grid, 4 → 2 → 1 columns across breakpoints
- Each `<li>` is a card with border, radius, and background from surface tokens
- On hover/focus-within: `translateY(-4px)`, stronger border glow, `h3` color shifts to `--primary`, body opacity increases
- `tabindex=0` on each `<li>` enables keyboard focus without JavaScript

Slideshow (`.attacks-slideshow`)

- Viewport: `display flex`, `overflow-x scroll`, `scroll-snap-type x mandatory`, `scrollbar-width none`
- Each slide: `flex-shrink 0`, `width 100%`, `scroll-snap-align start`
- Dots: small round buttons injected by JS into `.attacks-slideshow-dots`
- Prev/next: `.attacks-slideshow-btn` with SVG chevron icons

FAQ (`.faq-item`)

- `.faq-trigger`: full-width button, flex row, no default button chrome
- `.faq-chevron`: CSS-drawn chevron that rotates 180deg when `.is-open` is on the parent
- `.faq-panel`: `overflow hidden`, `grid-template-rows: 0fr`; transitions to `1fr` when `.is-open` is added
- `.faq-panel-inner`: `min-height 0` (required for grid row collapse trick)
- Motion: transition removed when `@media (prefers-reduced-motion: reduce)` matches

Contact Modal (`.contact-modal`)

- `dialog.contact-modal:not([open]) { display: none }` — prevents modal from flashing below footer
- `dialog.contact-modal[open]`: flex centering overlay
- `.contact-modal-panel`: max-width card with `radius-lg`, `surface-elevated` background
- Form grid: CSS Grid two-column on wide, single column on narrow; `textarea` spans full width
- At `≤540px`: panel anchored to bottom of viewport (bottom-sheet pattern)

4. JavaScript Modules

All six scripts are plain IIFE (immediately invoked function expression) or top-level modules. No transpilation, no bundler, no dependencies. Each script is self-contained and exits early if its required DOM element is not found.

4.1 js/banner.js — Hero Gradient Canvas

Draws a layered animated gradient on the full-bleed <canvas id="gradient-canvas"> behind the hero section.

Canvas Setup

- `resizeCanvas()` sets `canvas.width` and `canvas.height` to `canvas.offsetWidth / offsetHeight`
- Called on load and on the resize event (passive listener)

Gradient Blobs

Four blob objects, each with base x, y (0–1 normalized), radius r, and a hex color:

Blob	x	y	r	Color
0	0.2	0.3	0.9	#0d1741 (dark navy)
1	0.8	0.2	0.8	#5eead4 (teal)
2	0.6	0.8	0.9	#7c3aed (violet)
3	0.1	0.9	0.7	#1b3a5b (deep blue)

Animation

Each `draw()` frame applies sine/cosine offsets to each blob's center and radius:

- $\text{offsetX} = \sin(\text{time} \times 0.0012 + i \times 1.7) \times 0.16 + \cos(\text{time} \times 0.00055 + i) \times 0.05$
- $\text{offsetY} = \cos(\text{time} \times 0.001 + i \times 1.3) \times 0.14 + \sin(\text{time} \times 0.00065 + i \times 0.9) \times 0.05$
- $\text{radiusScale} = 1 + \sin(\text{time} \times 0.0009 + i \times 2.1) \times 0.14$
- Each blob is drawn as a `ctx.createRadialGradient` fading from its color to transparent
- Blobs are layered with `ctx.fillRect` — earlier blobs are painted first (additive blending)

Loop: `requestAnimationFrame(draw)` called each frame, passing a `DOMHighResTimeStamp`.

Reduced Motion

If `window.matchMedia('(prefers-reduced-motion: reduce)').matches`, `drawStatic()` is called once instead of starting the animation loop. `drawStatic()` renders the blobs at their base positions with no offsets.

4.2 js/attacks-slideshow.js — Carousel Controller

Initializes every `[data-iot-slideshow]` element on the page as an independent carousel. Both the Different Attacks carousel (12 slides, 6,500ms autoplay) and the How It Works carousel (10 slides, 7,000ms autoplay) use this same script.

Initialization

- `querySelectorAll('[data-iot-slideshow]')` finds all carousel roots

- Each root is initialized independently in a `forEach` loop — they share no state
- `data-autoplay-ms` attribute on the root sets the per-carousel autoplay interval (min 800ms; 0 or absent disables autoplay)
- Dots are created programmatically from the slides array and injected into `.attacks-slideshow-dots`

Navigation

- Prev/next buttons call `go(±1, true)` — scroll smooth
- Dot buttons call `scrollToIndex(j, true)`
- Keyboard: `ArrowLeft` / `ArrowRight` on the focused viewport, `preventDefault` to avoid page scroll

Scroll Sync

The viewport uses native CSS scroll-snap. The JS does not drive scrolling directly for touch/trackpad — instead:

- `viewport.addEventListener('scroll', scheduleReadScroll, { passive: true })`
- `scheduleReadScroll` debounces 120ms then calls `readIndexFromScroll()`
- `readIndexFromScroll()` calculates `index = Math.round(viewport.scrollLeft / viewport.clientWidth)`
- `syncChrome()` updates dot `aria-current` and slide `aria-hidden` to match

Autoplay

- `startAutoplay()` calls `window.setInterval(() => go(1, true), autoplayMs)`
- `stopAutoplay()` clears the interval
- Pauses: `mouseenter` on root (`pointerOverRoot = true`), `focusin` on root
- Resumes: `mouseleave` (if pointer left), `focusout` (only if pointer also not over root)
- Tab-visibility: `document.addEventListener('visibilitychange')` — stops on hidden, restarts on visible if pointer is not over root
- Disabled entirely: `prefers-reduced-motion` OR `autoplayMs < 800` OR `document.hidden` at init time

Resize Handling

- On resize: `viewport.scrollTo({ left: clientWidth * index, behavior: 'auto' })` to re-snap to current slide
- `syncChrome()` called after to keep dots correct

4.3 `js/faq.js` — FAQ Accordion

Single-open accordion for the 7-item FAQ. Finds the root by ID (faq-1193) then attaches click listeners to all `.faq-trigger` buttons.

Open / Close Logic

- On any trigger click, `wasOpen = item.classList.contains('is-open')`
- All items are closed: `.is-open` removed, `aria-expanded` set to `'false'` on all triggers
- If the clicked item was not already open: `.is-open` added back, `aria-expanded` set to `'true'`
- Clicking an open item closes it (toggle behavior)

CSS-Only Animation

- .faq-panel has grid-template-rows: 0fr and overflow: hidden
- When .is-open is on the parent .faq-item, .faq-panel gets grid-template-rows: 1fr
- CSS transition on grid-template-rows produces the smooth expand/collapse
- .faq-panel-inner has min-height: 0 — required for the grid row collapse to work
- transition disabled via @media (prefers-reduced-motion: reduce)

4.4 js/contact-modal.js — Contact Dialog

Controls the native <dialog id="contact-modal">. Requires the browser to support dialog.showModal() — exits early if not available.

Open

- openModal(): resets form, shows form + intro, hides success panel, sets aria-labelledby to contact-modal-title, calls dialog.showModal()
- Triggered by #contact-modal-open button click

Close

- closeModal(): guards dialog.open before calling dialog.close(); returns focus to the open button
- Triggered by: X close button (stopPropagation prevents backdrop close), success-state close button, or clicking the dialog backdrop
- Backdrop close: dialog.addEventListener('click', e => { if (e.target === dialog) closeModal() }) — works because the panel uses stopPropagation

Form Submit

- form.addEventListener('submit', ...) prevents default
- form.checkValidity() — if invalid, calls form.reportValidity() to show native browser validation UI
- On success: hides form and intro, shows success panel, updates aria-labelledby to contact-success-title
- Note: form data is not sent anywhere. A real backend, Formspree, or similar service must be connected before deployment

4.5 js/scroll-top.js — Back-to-Top Button

Controls the fixed round back-to-top button (#scroll-top) in the bottom-right corner.

Visibility

- syncVisibility() called on scroll (passive) and on load
- window.scrollY > 380: btn.classList.add('is-visible'), tabindex attribute removed
- window.scrollY ≤ 380: btn.classList.remove('is-visible'), tabindex set to '-1'
- tabindex=-1 when hidden so keyboard Tab order skips the button

Click

- `window.scrollTo({ top: 0, behavior: 'smooth' })`
- behavior falls back to 'auto' if `prefers-reduced-motion: reduce` is set

4.6 js/section-nav.js — Sticky Section Navigation

Controls the fixed dot-navigation rail (`#section-nav`) that appears on the right side on wide viewports. Also manages the hero panel departure animation.

Hero Departure

- `#hero-learn-more` click: `heroPanel.classList.add('is-departing')` — triggers CSS lift + fade
- Scroll listener: if `window.scrollY < 72`, removes `is-departing` (resets when user scrolls back to top)

Viewport Guard

- If `!window.matchMedia('(min-width: 1320px)').matches`, the rest of the script returns immediately
- The nav is CSS-hidden below 1320px, so no scroll-spy runs on mobile or tablet

Reveal on Scroll

- `pastHero()`: `hero.getBoundingClientRect().bottom < 48`
- `setRevealed()`: toggles `nav.classList 'is-revealed'` and `nav.setAttribute('aria-hidden', ...)` based on `pastHero()`
- When not past hero: all links lose `.is-active` and `aria-current`

Active Section Tracking

- `updateFromScroll()` is called on every scroll event (passive)
- At bottom of page: last section is always set active
- Otherwise: `mark = window.innerHeight * 0.32` (32% from top of viewport)
- Loops through all tracked section IDs; sets current to any whose `getBoundingClientRect().top <= mark`
- `setActive(id)`: toggles `.is-active` and `aria-current='location'` on the matching nav link
- Links also call `requestAnimationFrame(updateFromScroll)` on click for immediate active-state update

5. Images & Assets

All images live in `assets/images/`. They are referenced directly in `index.html` — no lazy-loading library is used; HTML native `loading="lazy"` and `decoding="async"` attributes are set on all non-hero images.

5.1 Architecture & Diagram Images

Explanation.png	System architecture overview diagram
logflow.jpeg	Log pipeline from IoT devices to SIEM
siemLab.jpeg	Full SIEM lab topology
teamroles.jpeg	Developer roles and responsibilities
workflow.jpeg	Threat simulation and alerting workflow
SixSprint.jpeg	6-sprint Agile development timeline
Outcomes.jpeg	Project outcomes and results summary
iot-shield-system.jpg	System architecture photo (used in Our Solution section)
thinking_computer.jpg	Decorative image in FAQ section (aria-hidden)

5.2 Dashboard Screenshots

IoT_Shield_-_Dashboard.png	Main Operations dashboard: event counts, time-series chart, top devices
IoT_Shield_-_Security_Alerts.png	Security Alerts dashboard: 900+ alerts, alert-type breakdown bars
Malware_Signature_Detected.png	Splunk search: malware signature results (used in both carousels)

5.3 Attack Example Screenshots (Different Attacks carousel)

Brute_Force_Attack_-_Failed_Logins.png	Brute force: repeated failed authentication
Data_Exfiltration_-_Large_Outbound_Transfer.png	Data exfiltration: large outbound transfer
Directory_Traversal_Attack_Attempt.png	Directory traversal attempt
Port_Scan_Detection_-_Multiple_Ports.png	Port scan across multiple ports
Possible_Denial_of_Service_Attack.png	Possible denial of service
Privilege_Escalation_Attempts_Detected.png	Privilege escalation attempts
Sensitive_Data_in_Outbound_Traffic.png	Sensitive data in outbound traffic
SQL_Injection_Attack_Detected.png	SQL injection detected

Unauthorized Service Access Attempts.png	Unauthorized service access
XSS Attack Pattern Detected.png	Cross-site scripting (XSS) pattern

All carousel images are 960 × 540px. width and height attributes are set in the HTML to prevent layout shift. Images with spaces in their filename (Unauthorized Service Access Attempts.png, XSS Attack Pattern Detected.png) are referenced with the space character in the src attribute — no URL encoding is applied in index.html.

6. Extending & Maintaining the Site

6.1 Adding a Carousel Slide

To add a slide to either carousel, insert an `article.attacks-slide` inside the `.attacks-slideshow-viewport` before the closing `</div>`:

```
<article class="attacks-slide" data-label="Your slide label">
  <figure class="attacks-slide-figure">
    
    <figcaption class="attacks-slide-caption">Caption text</figcaption>
  </figure>
</article>
```

The JS will automatically pick up the new slide, generate an additional dot button, and include it in the aria-live status count. No JavaScript changes needed.

6.2 Adding a FAQ Item

Inside the `<ul class="faq-group">`, add a new `<li>` following this pattern:

```
<li class="faq-item">
  <button type="button" class="faq-trigger" id="faq-btn-8"
    aria-expanded="false" aria-controls="faq-panel-8">
    <span class="faq-question">Your question here?</span>
    <span class="faq-chevron" aria-hidden="true"></span>
  </button>
  <div class="faq-panel" id="faq-panel-8" role="region" aria-labelledby="faq-btn-8">
    <div class="faq-panel-inner">
      <p class="faq-answer">Your answer here.</p>
    </div>
  </div>
</li>
```

No JavaScript changes needed. The `faq.js` click handler uses `querySelectorAll('.faq-trigger')` and will automatically include the new item.

6.3 Adding a Feature Card

Inside `.sf-product-tab-order-list`, copy an existing `<li>` and update the number, heading, and description. The CSS grid automatically repositions cards at each breakpoint.

6.4 Wiring the Contact Form

The form has `id="contact-form"` and `name="Contact Form"`. To connect it to a real backend:

- Formspree: add `action="https://formspree.io/f/YOUR_ID"` `method="POST"` to the `<form>` tag and remove the JS submit handler's `e.preventDefault()` guard.

- Custom API: in contact-modal.js, replace the success-state swap logic with a fetch() POST to your endpoint. Show the success panel on a resolved promise.
- Netlify Forms: add netlify attribute to the <form> tag; Netlify detects and handles it automatically during deployment.

6.5 Changing Autoplay Speed

Each carousel reads its own data-autoplay-ms attribute. Find the root div and update the value:

```
<!-- Different Attacks carousel -->
<div class="attacks-slideshow" data-iot-slideshow data-autoplay-ms="6500">

<!-- How it works carousel -->
<div class="attacks-slideshow" data-iot-slideshow data-autoplay-ms="7000">
```

Set to 0 or remove the attribute entirely to disable autoplay on a specific carousel.

6.6 Changing the Hero Gradient Colors

In js/banner.js, the colors array contains the four blob definitions. Update the color hex values there. The gradient is canvas-drawn — no CSS change needed.

7. Known Limitations

- Contact form is a client-side demo only. Form data is not submitted anywhere. A backend integration is required before deployment.
- No automated tests. All functionality was verified through manual browser testing across Chrome, Firefox, and Safari.
- Images with spaces in filenames (Unauthorized Service Access Attempts.png, XSS Attack Pattern Detected.png) should ideally be renamed with underscores for cross-platform compatibility, though current references work in all tested browsers.
- Large PNGs in assets/images/ are not compressed or converted to WebP. Consider running them through an image optimizer (e.g., Squoosh, ImageMagick) before production deployment to improve Lighthouse performance score.
- The section nav only activates above 1320px. There is no mobile equivalent navigation component.
- The hero canvas animation uses requestAnimationFrame continuously. On battery-constrained devices this may increase power usage; the prefers-reduced-motion branch mitigates this for users who have opted into reduced motion.
- Google Fonts requires outbound internet access. For a fully offline or intranet deployment, download and self-host the Outfit and DM Sans font files and update the CSS @font-face declarations.