Note: Please only Copy the Blue color text

#Extract Headers for Debug Node:

```
// Format headers into a readable string
let formattedHeaders = '';
if (items[0].json.headers) {
  for (const key in items[0].json.headers) {
    formattedHeaders += `${key}: ${items[0].json.headers[key]}\n`;
  }
}
// Return both the original data and the formatted headers
return [{
  json: {
    ...items[0].json,
    formattedHeaders: formattedHeaders,
    originalHeaders: items[0].json.headers // Keep the original headers too
  }
}];
```

#Security Configuration Audit Node (Prompt User Message):

You are an elite web security expert specializing in secure configurations.

In this task, you will analyze the HTTP headers, cookies, and overall configuration of a webpage to identify security misconfigurations.

Audit Structure

You will begin by listing ALL security headers that ARE present and properly configured.

Be very clear and explicit about which headers are present and which are missing. For each header, clearly state whether it is present or missing, and if present, what its value is.

Then, present your findings in three sections:

- Header Security – Missing or misconfigured security headers

- Cookie Security – Insecure cookie configurations

- Content Security – CSP issues, mixed content, etc.

For each finding, provide:

1. A clear description of the misconfiguration

2. The security implications

3. The recommended secure configuration with example code

If you find no issues in a particular section, explicitly state that no issues were found.

Use proper formatting with code blocks for configuration examples. Only include issues that can be detected from client-side inspection.

Here are the response headers: {{ $json.formattedHeaders }}

Please Respond like this

[any section heading that includes "Headers]

1. **[Header Title]**

   - **Present?** Yes/No

   - **Value:** `actual-header-value`

#Security Vulnerabilities Audit Node (Prompt User Message):

You are an elite cybersecurity expert specializing in web application security.

In this task, you will analyze the HTML and visible content of the webpage to identify potential security vulnerabilities.

Audit Structure

You will review all client-side security aspects of the page and present your findings in three sections:

- Critical Vulnerabilities – Issues that could lead to immediate compromise

- Information Leakage – Sensitive data exposed in page source

- Client-Side Weaknesses – JavaScript vulnerabilities, XSS opportunities, etc.

For each issue found, provide:

1. A clear description of vulnerability

2. The potential impact

3. A specific recommendation to fix it

If you find no issues in a particular section, explicitly state that no issues were found in that category.

Ensure the output is properly formatted, clean, and highly readable. Focus only on issues that can be detected from the client-side code.

Here is the content of the webpage: {{ $json.test }}

Process Audit Results Code Node:

```
// ✅ Updated extractSecurityHeaders and related logic remains unchanged
function extractSecurityHeaders(rawHeaders = {}, configOutput = '') {
  const securityHeaders = [
    'Content-Security-Policy',
    'Strict-Transport-Security',
    'X-Content-Type-Options',
    'X-Frame-Options',
    'Referrer-Policy',
    'Permissions-Policy',
    'X-XSS-Protection',
    'Cross-Origin-Embedder-Policy',
    'Cross-Origin-Opener-Policy',
    'X-Permitted-Cross-Domain-Policies'
  ];
  const headerStatus = {};
  for (const header of securityHeaders) {
    headerStatus[header] = { present: false, value: '' };
  }
  for (const header in rawHeaders) {
    const norm = header.trim().toLowerCase();
    for (const standard of securityHeaders) {
      if (norm === standard.toLowerCase()) {
        headerStatus[standard].present = true;
        headerStatus[standard].value = rawHeaders[header];
      }
    }
  }
```

```
  const presentSection =
configOutput.match(/(?:###|##|\*\*)[^\n]*?\bheaders?\b[\s\S]*?(?=###|##|\*\*|$)/i);
  if (presentSection) {
    const section = presentSection[0];
    for (const header of securityHeaders) {
      const title = header.replace(/-/g, ' ').replace(/\b\w/g, c => c.toUpperCase());
      const regex = new
RegExp(`\\*\\*${title}\\*\\*[^\\n]*?\\*\\*Present\\?\\*\\*\\s*Yes[^\\n]*?\\*\\*Value:\\*\\*\\s*\`([^\\\`]+)\``, 'is');
      const match = section.match(regex);
      if (match && match[1]) {
        headerStatus[header].present = true;
        headerStatus[header].value = match[1].trim();
      }
    }
  }
  return headerStatus;
}
function hasUnsafeInline(value) {
  return value && value.includes('unsafe-inline');
}
function determineGrade(headerStatus) {
  const critical = [
    'Content-Security-Policy',
    'Strict-Transport-Security',
    'X-Content-Type-Options',
    'X-Frame-Options'
  ];
  const important = ['Referrer-Policy', 'Permissions-Policy'];
  const additional = [
    'X-XSS-Protection',
```

```
  'Cross-Origin-Embedder-Policy',
  'Cross-Origin-Opener-Policy',
  'X-Permitted-Cross-Domain-Policies'
];
let criticalCount = 0;
let importantCount = 0;
let hasCSPIssue = false;
for (const h of critical) {
  if (headerStatus[h]?.present) {
    criticalCount++;
    if (h === 'Content-Security-Policy' && hasUnsafeInline(headerStatus[h].value)) {
      hasCSPIssue = true;
    }
  }
}
for (const h of important) {
  if (headerStatus[h]?.present) importantCount++;
}
if (criticalCount === critical.length) {
  if (importantCount === important.length) return hasCSPIssue ? 'A-' : 'A+';
  if (importantCount >= 1) return hasCSPIssue ? 'B+' : 'A-';
  return hasCSPIssue ? 'B' : 'B+';
} else if (criticalCount >= critical.length - 1) {
  return importantCount >= 1 ? 'B' : 'C+';
} else if (criticalCount >= 2) {
  return 'C';
} else if (criticalCount >= 1) {
  return 'D';
} else {
```

```
    return 'F';
  }
}
function formatHeadersForDisplay(headerStatus) {
  const present = Object.keys(headerStatus).filter(h => headerStatus[h].present);
  return present.length > 0 ? present.join(', ') : 'No security headers detected';
}
function processSecurityHeaders(items) {
  try {
    const json = items[0].json || items[0];
    // ⛏️ Try to grab from originalHeaders if available
    const rawHeaders =
      json?.originalHeaders ||
      $('Extract Headers for Debug')?.first()?.json?.originalHeaders ||
      json?.headers ||
      {};
    const configOutput = json.configOutput || json.output?.[0] || '';
    const vulnOutput = json.vulnOutput || json.output?.[1] || '';
    const headerStatus = extractSecurityHeaders(rawHeaders, configOutput);
    const presentHeaders = formatHeadersForDisplay(headerStatus);
    const grade = determineGrade(headerStatus);
    const timestamp = new Date().toLocaleString('en-US', {
      year: 'numeric',
      month: 'long',
      day: 'numeric',
      hour: '2-digit',
      minute: '2-digit'
    });
    const url =
```

```
  json?.formValues?.url ||
  json?.['Landing Page Url'] ||
  $('Landing Page Url')?.first()?.json?.['Landing Page Url'] ||
  json?.Landing_Page_Url ||
  json?.landingPageUrl ||
  json?.url ||
  'https://example.com';
return [
  {
    json: {
      ...json,
      auditData: {
        url,
        timestamp,
        grade,
        criticalCount:
          headerStatus['Content-Security-Policy'].present &&
          hasUnsafeInline(headerStatus['Content-Security-Policy'].value)
            ? 1
            : 0,
        warningCount: Object.keys(headerStatus).filter(
          h =>
            !headerStatus[h].present &&
            !['Strict-Transport-Security', 'Content-Security-Policy'].includes(h)
        ).length,
        presentHeaders,
        configOutput,
        vulnOutput,
        headerStatus,
```

```
            originalHeaders: rawHeaders
          }
        }
      }
    ];
  } catch (err) {
    return [{ json: { ...items[0].json, error: err.message } }];
  }
}
return processSecurityHeaders(items);
```

convert to HTML Code Node:

```
// Create a direct HTML template with improved styling
const auditData = items[0].json.auditData;

function formatConfigurationIssues() {
  if (!auditData.configOutput || auditData.configOutput.trim() === '') {
    return '<p>No specific configuration issues detected.</p>';
  }

  try {
    const config = auditData.configOutput.trim();
    let html = '';
    const renderedKeys = new Set();

    const renderBlock = (title, description, impact, recommendation) => `
      <div style="border-left: 4px solid #3498DB; padding: 10px; margin-bottom: 15px;">
        <div style="font-weight: bold; color: #3498DB;">${title}</div>
        ${description ? `<div style="margin-top: 5px;">${description}</div>` : ''}
```

```
      ${impact ? `<div style="margin-top: 5px; font-style: italic; color: #7F8C8D;">Impact: ${impact}</div>` : ''}
      ${recommendation ? `<div style="margin-top: 5px;"><strong>Recommendation:</strong></div>
        <pre style="background-color: #f8f9fa; padding: 10px; border-radius: 5px; overflow-x: auto; font-family: monospace;">${recommendation}</pre>` : ''}
    </div>`;

  const sections = config.split(/(?=^###\s+)/gm).filter(Boolean);

  for (const section of sections) {
    const sectionTitleMatch = section.match(/^###\s+(.*)/);
    const sectionTitle = sectionTitleMatch?.[1]?.trim() || 'Unnamed Section';
    const sectionKey = sectionTitle.toLowerCase();

    // Skip "no issues found" sections
    if (/no issues? (found|were found)/i.test(section)) continue;

    const lines = section.split(/\n+/).filter(line => line.trim() !== '');

    let currentTitle = '';
    let description = '';
    let impact = '';
    let recommendation = '';

    for (let i = 0; i < lines.length; i++) {
      const line = lines[i].trim();

      // Start of a new numbered or bolded issue
      const numberedTitle = line.match(/^\d+\.\s+\*\*(.*?)\*\*/);
      const bulletTitle = line.match(/^\*\*(.*?)\*\*/);
```

```
if (numberedTitle || (!currentTitle && bulletTitle)) {
  // Flush last block
  if (currentTitle && !renderedKeys.has(`${sectionKey}::${currentTitle.toLowerCase()}`)) {
    html += renderBlock(currentTitle, description, impact, recommendation);
    renderedKeys.add(`${sectionKey}::${currentTitle.toLowerCase()}`);
  }

  currentTitle = (numberedTitle || bulletTitle)[1].trim();
  description = '';
  impact = '';
  recommendation = '';
  continue;
}

const valueMatch = line.match(/- \*\*Value:\*\*\s*`?(.*?)`?$/i);
const presentMatch = line.match(/- \*\*Present\?\*\*.*?(Yes|No)/i);
const descMatch = line.match(/- \*\*Description:\*\*\s*(.*)/i);
const impactMatch = line.match(/- \*\*(?:Impact|Security Implication|Potential Impact):\*\*\s*(.*)/i);
const recMatch = line.match(/```(?:\w*)?\n([\s\S]*?)```/i);

if (descMatch) {
  description = descMatch[1].trim();
} else if (valueMatch || presentMatch) {
  const present = presentMatch?.[1]?.trim() || 'Unknown';
  const value = valueMatch?.[1]?.trim() || '[Not provided]';
  description = `This header is ${present.toLowerCase()}. Value: ${value}.`;
}
```

```
        if (impactMatch) {
          impact = impactMatch[1].trim();
        }

        if (recMatch) {
          recommendation = recMatch[1].trim();
        }
      }

      // Final block in section
      if (currentTitle && !renderedKeys.has(`${sectionKey}::${currentTitle.toLowerCase()}`)) {
        html += renderBlock(currentTitle, description, impact, recommendation);
        renderedKeys.add(`${sectionKey}::${currentTitle.toLowerCase()}`);
      }
    }

    return html || '<p>No configuration issues detected.</p>';
  } catch (e) {
    console.error('Error in formatConfigurationIssues:', e);
    return `<p>Error processing configuration issues: ${e.message}</p>`;
  }
}



// Create header badge HTML
function createHeaderBadge(headerName, isWarning = false) {
  const isPresent = auditData.headerStatus &&
                    auditData.headerStatus[headerName] &&
```

```
                auditData.headerStatus[headerName].present;

  const color = isWarning && isPresent ? "#F39C12" : (isPresent ? "#27AE60" : "#E74C3C");
  const icon = isPresent ? "✓" : "✗";

  return `<span style="display: inline-block; margin: 2px; padding: 4px 8px; background-color: ${color};
color: white; border-radius: 4px; font-size: 12px;">${icon} ${headerName}</span>`;
}

// Format warnings section
function formatWarningsSection() {
  if (!auditData.warningCount || auditData.warningCount === 0 || !auditData.headerStatus) {
    return '<p>No warnings detected.</p>';
  }

  const csp = Object.entries(auditData.headerStatus).find(([k]) => k.toLowerCase() ===
'content-security-policy');
  const hsts = Object.entries(auditData.headerStatus).find(([k]) => k.toLowerCase() ===
'strict-transport-security');
  const xss = Object.entries(auditData.headerStatus).find(([k]) => k.toLowerCase() ===
'x-xss-protection');

  let warnings = '';

  if (csp && csp[1].value && csp[1].value.includes('unsafe-inline')) {
    warnings += `
      <div style="margin-top: 15px;">
        <div style="border-left: 4px solid #F39C12; padding: 10px;">
          <strong style="color: #F39C12;">Content-Security-Policy: unsafe-inline</strong>
          <p>The use of 'unsafe-inline' allows potentially malicious scripts to execute.</p>
```

```
      </div>
    </div>`;
}

if (hsts && hsts[1].value) {
  const match = hsts[1].value.match(/max-age=(\d+)/);
  const age = match ? parseInt(match[1]) : 0;
  if (age < 2592000) {
    warnings += `
      <div style="margin-top: 15px;">
        <div style="border-left: 4px solid #F39C12; padding: 10px;">
          <strong style="color: #F39C12;">Strict-Transport-Security</strong>
          <p>max-age is too low (${age}). Should be at least 2592000 (30 days).</p>
        </div>
      </div>`;
  }
}

if (xss && !xss[1].present) {
  warnings += `
    <div style="margin-top: 15px;">
      <div style="border-left: 4px solid #F39C12; padding: 10px;">
        <strong style="color: #F39C12;">Missing X-XSS-Protection</strong>
        <p>This header enables the browser's XSS filter. Lack of it increases XSS risks.</p>
      </div>
    </div>`;
}

if (!warnings) {
```

```
    warnings = `
      <div style="margin-top: 15px;">
        <div style="border-left: 4px solid #F39C12; padding: 10px;">
          <strong style="color: #F39C12;">${auditData.warningCount} warnings detected</strong>
          <p>See the Configuration Issues section below for more info.</p>
        </div>
      </div>`;
  }

  return warnings;
}

function formatLongValue(value) {
  if (!value || typeof value !== 'string') return '[empty]';

  // Convert URLs into clickable links
  value = value.replace(/(https?:\/\/[^\s]+)/g, '<a href="$1" style="color: #3498DB; text-decoration: none;" target="_blank">$1</a>');

  // Add line breaks after commas or semicolons for readability
  if (value.length > 100) {
    value = value.replace(/([,;])\s*/g, '$1<br>');
  }

  return value;
}

function formatDetailedRawHeaders() {
  const allHeaders = [];
```

```
const seen = new Set();

const addHeader = (name, value) => {
  const key = name.toLowerCase();
  if (seen.has(key)) return;
  seen.add(key);

  const status = Object.entries(auditData.headerStatus || {}).find(
    ([k]) => k.toLowerCase() === name.toLowerCase()
  );
  const present = status ? status[1].present : !!value;

  allHeaders.push({
    name: name.trim(),
    present,
    value: value || '[empty]'
  });
};

Object.entries(auditData.originalHeaders || {}).forEach(([key, value]) => {
  if (key) addHeader(key, value);
});

const securityHeaders = [
  'content-security-policy',
  'strict-transport-security',
  'x-content-type-options',
  'x-frame-options',
  'referrer-policy',
```

```
  'permissions-policy',
  'x-xss-protection'
];

const isWarningHeader = (name, value) => {
  const lower = name.toLowerCase();
  if (lower === 'strict-transport-security') {
    const match = value.match(/max-age=(\d+)/);
    return match && parseInt(match[1]) < 2592000;
  }
  if (lower === 'content-security-policy') return value.includes("'unsafe-inline'");
  return false;
};

const tableRows = allHeaders.map(header => {
  const isSecurity = securityHeaders.includes(header.name.toLowerCase());
  const warning = isSecurity && isWarningHeader(header.name, header.value);
  const missing = isSecurity && !header.present;

  let bgColor = '#F8F9FA';
  let textColor = '#333';

  if (isSecurity) {
    if (missing) {
      bgColor = '#FFEBEE';
      textColor = '#C62828';
    } else if (warning) {
      bgColor = '#FFF9C4';
      textColor = '#F57F17';
```

```
      } else {
        bgColor = '#E8F5E9';
        textColor = '#2E7D32';
      }
    }

    return `
      <tr style="background-color: ${bgColor}; color: ${textColor};">
        <td title="${isSecurity ? (missing ? 'Missing' : (warning ? 'Needs review' : 'Secure')) : 'Informational'}" style="padding: 8px; font-weight: bold;">${header.name}</td>
        <td style="padding: 8px; text-align: center;">${header.present ? 'present' : 'absent'}</td>
        <td style="padding: 8px; word-break: break-word; font-family: monospace;">${formatLongValue(header.value)}</td>
      </tr>`;
  }).join('');

  return `
    <table style="width: 100%; border-collapse: collapse; margin-top: 10px;">
      <thead>
        <tr style="background-color: #E0E0E0;">
          <th style="padding: 10px;">Header</th>
          <th style="padding: 10px;">Status</th>
          <th style="padding: 10px;">Value</th>
        </tr>
      </thead>
      <tbody>
        ${tableRows}
      </tbody>
    </table>`;
}
```

```
// Format additional information section
function formatAdditionalInfo() {
  const headers = [
    {
      name: 'access-control-allow-origin',
      description: 'This is a very lax CORS policy. Such a policy should only be used on a public CDN.'
    },
    {
      name: 'strict-transport-security',
      description: 'HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS.'
    },
    {
      name: 'content-security-policy',
      description: 'Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets. Analyse this policy in more detail. You can sign up for a free account on Report URI to collect reports about problems on your site.'
    },
    {
      name: 'permissions-policy',
      description: 'Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.'
    },
    {
      name: 'referrer-policy',
      description: 'Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.'
    },
    {
```

```
    name: 'x-content-type-options',
    description: 'X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".'
  },
  {
    name: 'x-frame-options',
    description: 'X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.'
  },
  {
    name: 'report-to',
    description: 'Report-To enables the Reporting API. This allows a website to collect reports from the browser about various errors that may occur. You can sign up for a free account on Report URI to collect these reports.'
  },
  {
    name: 'nel',
    description: 'Network Error Logging is a new header that instructs the browser to send reports during various network or application errors. You can sign up for a free account on Report URI to collect these reports.'
  },
  {
    name: 'server',
    description: 'Server value has been changed. Typically you will see values like "Microsoft-IIS/8.0" or "nginx 1.7.2".'
  }
];

let rows = '';
```

```
  for (const header of headers) {
    const isSecurityHeader = ['content-security-policy', 'strict-transport-security', 'x-content-type-options', 'x-frame-options', 'referrer-policy', 'permissions-policy'].includes(header.name);
    const headerColor = isSecurityHeader ? '#27AE60' : '#3498DB';

    rows += `
      <tr>
        <td style="padding: 8px; border-bottom: 1px solid #eee; color: ${headerColor}; font-weight: bold;">${header.name}</td>
        <td style="padding: 8px; border-bottom: 1px solid #eee;">${header.description}</td>
      </tr>
    `;
  }

  return `
    <table style="width: 100%; border-collapse: collapse; margin-top: 10px;">
      <tbody>
        ${rows}
      </tbody>
    </table>
  `;
}

function formatSecurityGrade() {
  const gradeColors = {
    'A+': '#27AE60',
    'A': '#27AE60',
    'A-': '#27AE60',
    'B+': '#3498DB',
```

```
    'B': '#3498DB',
    'B-': '#3498DB',
    'C+': '#F39C12',
    'C': '#F39C12',
    'C-': '#F39C12',
    'D+': '#E74C3C',
    'D': '#E74C3C',
    'D-': '#E74C3C',
    'F': '#E74C3C'
  };

  return `<div class="grade" style="font-size: 64px; font-weight: bold; width: 100px; height: 100px; line-height: 100px; text-align: center; background-color: ${gradeColors[auditData.grade] || '#E74C3C'}; color: white; border-radius: 5px; margin: 0 auto;">${auditData.grade}</div>`;
}

function formatCriticalVulnerabilities() {
  if (!auditData.vulnOutput || auditData.vulnOutput.trim() === '') {
    return '<p>No vulnerabilities detected.</p>';
  }

  try {
    const vuln = auditData.vulnOutput.trim();
    let html = '';
    const renderedTitles = new Set();

    // Match sections like ## Category (e.g., ## Critical Vulnerabilities)
    const categories = vuln.split(/(?=^##\s+)/gm).filter(Boolean);

    for (const categoryBlock of categories) {
```

```
    const categoryMatch = categoryBlock.match(/^##\s+(.*)/);
    const categoryTitle = categoryMatch?.[1]?.trim() || 'Uncategorized';

    // Find numbered items: 1. **Title**
    const vulns = categoryBlock.split(/(?=^\d+\.\s+\*\*)/gm).filter(Boolean);

    for (const vulnBlock of vulns) {
      const titleMatch = vulnBlock.match(/^\d+\.\s+\*\*(.*?)\*\*/);
      const title = titleMatch?.[1]?.trim() || 'Unnamed Vulnerability';
      const key = `${categoryTitle}::${title}`.toLowerCase();
      if (renderedTitles.has(key)) continue;

      const descriptionMatch = vulnBlock.match(/\*\*Description\*\*:?\s*([\s\S]*?)(?=\n\*\*|\n$)/i);
      const impactMatch = vulnBlock.match(/\*\*(?:Impact|Potential Impact)\*\*:?\s*([\s\S]*?)(?=\n\*\*|\n$)/i);
      const recommendationMatch = vulnBlock.match(/\*\*(?:Recommendation|Mitigation|Fix)\*\*:?\s*([\s\S]*?)(?=\n\*\*|\n$)/i);

      const description = descriptionMatch?.[1]?.trim() || '';
      const impact = impactMatch?.[1]?.trim() || '';
      const recommendation = recommendationMatch?.[1]?.trim() || '';

      if (description || impact || recommendation) {
        html += `
          <div style="border-left: 4px solid #E74C3C; padding: 10px; margin-bottom: 15px;">
            <div style="font-weight: bold; color: #E74C3C;">${title}</div>
            ${description ? `<div style="margin-top: 5px;">${description}</div>` : ''}
            ${impact ? `<div style="margin-top: 5px; font-style: italic; color: #7F8C8D;">Impact: ${impact}</div>` : ''}
```

```
            ${recommendation ? `<div style="margin-top: 5px;"><strong>Recommendation:</strong>
${recommendation}</div>` : ''}
          </div>`;
        renderedTitles.add(key);
      }
    }
  }

  return html || '<p>No vulnerabilities parsed from output.</p>';
 } catch (e) {
  console.error('Error in formatCriticalVulnerabilities:', e);
  return `<p>Error processing vulnerabilities: ${e.message}</p>`;
 }
}


// Generate all security header badges
function generateAllHeaderBadges() {
 // Only include the necessary security headers
 const securityHeaders = [
  'Content-Security-Policy',
  'Strict-Transport-Security',
  'X-Content-Type-Options',
  'X-Frame-Options',
  'Referrer-Policy',
  'Permissions-Policy'
 ];

 let badges = '';
```

```
  securityHeaders.forEach(header => {

    const isWarning = header === 'Strict-Transport-Security' &&
                  auditData.headerStatus?.[header]?.value &&
                  parseInt(auditData.headerStatus[header].value.match(/max-age=(\d+)/)?.[1] || 0) <
2592000;

    badges += createHeaderBadge(header, isWarning);
  });

  return badges;
}

<!-- Modify the HTML to directly access auditData.originalHeaders or allHeaders -->
const html = `<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Website Security Audit Report</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f9f9f9; }
        .container { max-width: 950px; margin: 0 auto; }
        .header { background-color: #2c3e50; color: white; padding: 25px 20px; text-align: center; }
        .header h1 { color: white; font-size: 28px; margin: 0; text-shadow: 1px 1px 2px rgba(0,0,0,0.5); }
        .content { padding: 20px; }
        .summary-box { background-color: #EBF5FB; padding: 15px; margin-bottom: 20px;
border-radius: 5px; box-shadow: 0 1px 3px rgba(0,0,0,0.1); }
        .warning-box { background-color: #FEF5E7; padding: 15px; margin-bottom: 20px; border-radius:
5px; box-shadow: 0 1px 3px rgba(0,0,0,0.1); }
```

```
        .headers-box { background-color: #F5F7FA; padding: 15px; margin-bottom: 20px; border-radius: 5px; }
        .findings-box { background-color: white; padding: 15px; margin-bottom: 20px; border-radius: 5px; box-shadow: 0 1px 3px rgba(0,0,0,0.1); }
        .raw-headers-box { background-color: #F5F7FA; padding: 15px; margin-bottom: 20px; border-radius: 5px; box-shadow: 0 1px 3px rgba(0,0,0,0.1); }
        .additional-info-box { background-color: #F5F7FA; padding: 15px; margin-bottom: 20px; border-radius: 5px; box-shadow: 0 1px 3px rgba(0,0,0,0.1); }
        .details-table { width: 100%; border-collapse: collapse; }
        .details-table th { text-align: left; padding: 8px; background-color: #f2f2f2; }
        .details-table td { padding: 8px; border-bottom: 1px solid #eee; }
        .header-badges { margin-top: 10px; }
        h1, h2, h3 { color: #2c3e50; }
        .critical-item { border-left: 4px solid #E74C3C; padding: 10px; margin-bottom: 15px; }
        .critical-title { font-weight: bold; color: #E74C3C; }
        .config-item { border-left: 4px solid #3498DB; padding: 10px; margin-bottom: 15px; }
        .config-title { font-weight: bold; color: #3498DB; }
        pre { background-color: #f8f9fa; padding: 10px; border-radius: 5px; overflow-x: auto; font-family: monospace; margin-top: 5px; }
    </style>
</head>
<body>
    <div class="container">
        <!-- Report Header -->
        <div class="header">
            <h1 style="color: white; text-shadow: 1px 1px 2px rgba(0,0,0,0.5);">Website Security Audit Report</h1>
        </div>

        <div class="content">
            <!-- Security Report Summary -->
```

```
        <div class="summary-box">
            <h2>Security Report Summary</h2>
            <table style="width: 100%;">
                <tr>
                    <td style="width: 120px;" valign="top">
                        ${formatSecurityGrade()}
                    </td>
                    <td valign="top">
                        <table style="width: 100%;">
                            <tr>
                                <td><strong>Site:</strong></td>
                                <td><a href="${auditData.url}" style="color:
#3498db;">${auditData.url}</a></td>
                            </tr>
                            <tr>
                                <td><strong>Report Time:</strong></td>
                                <td>${auditData.timestamp}</td>
                            </tr>
                            <tr>
                                <td valign="top"><strong>Headers:</strong></td>
                                <td>
                                    <div class="header-badges">
                                        ${generateAllHeaderBadges()}
                                    </div>
                                </td>
                            </tr>
                            <tr>
                                <td><strong>Critical Issues:</strong></td>
                                <td>${auditData.criticalCount || 0}</td>
```

```
                </tr>
                <tr>
                    <td><strong>Warnings:</strong></td>
                    <td>${auditData.warningCount || 0}</td>
                </tr>
            </table>
        </td>
    </tr>
</table>
</div>

<!-- Warnings Section -->
<div class="warning-box">
    <h2>Warnings</h2>
    ${formatWarningsSection()}
</div>

<!-- Raw Headers Section -->
<div class="raw-headers-box">
    <h2>Raw Headers</h2>
    ${formatDetailedRawHeaders()}
</div>

<!-- Security Findings -->
<div class="findings-box">
    <h2>Security Findings</h2>

    <!-- Vulnerabilities -->
    <h3>Vulnerabilities</h3>
```

```
        ${formatCriticalVulnerabilities()}

        <!-- Configuration Issues -->
        <h3>Configuration Issues</h3>
        ${formatConfigurationIssues()}
      </div>

      <div class="additional-info-box">
        <h2>Additional Information</h2>
        ${formatAdditionalInfo()}
      </div>
            <!-- Implementation Guide -->
      <div class="findings-box">
        <h2>Implementation Guide</h2>
        <p>This report highlights security issues detected through client-side analysis. For a comprehensive security assessment, consider engaging a professional penetration tester.</p>

        <div style="background-color: #eafaf1; padding: 15px; margin-top: 15px; border-left: 4px solid #2ecc71; border-radius: 3px;">
          <p><strong>To implement the fixes above:</strong></p>
          <ol style="padding-left: 20px; margin-top: 10px;">
            <li>Work with your development team to address each issue in order of criticality</li>
            <li>Retest after implementing each fix</li>
            <li>Consider implementing a web application firewall for additional protection</li>
          </ol>
        </div>
      </div>

      <!-- Footer -->
      <div style="text-align: center; padding: 20px; font-size: 12px; color: #777;">
```

```
            <p>This report was generated by AI University Pakistan’s Automated Security Intelligence System.
It offers a rapid assessment of publicly accessible elements of your digital infrastructure.
For in-depth cybersecurity audits or AI-powered consulting, connect with our expert team.</p>
            <p>&copy; 2025 AI University Pakistan | Generated on ${auditData.timestamp} </p>
          </div>
        </div>
    </div>
</body>
</html>`;
return [{
  json: {
    ...items[0].json,
    emailHtml: html
  }
}];
```