What Is Mixed Content?

By Jo-el van Bergen
Jo-el is a contributor to Web**Fundamentals**

Mixed content occurs when initial HTML is loaded over a secure HTTPS connection, but other resources (such as images, videos, stylesheets, scripts) are loaded over an insecure HTTP connection. This is called mixed content because both HTTP and HTTPS content are being loaded to display the same page, and the initial request was secure over HTTPS. Modern browsers display warnings about this type of content to indicate to the user that this page contains insecure resources.

TL;DR

- HTTPS is important to protect both your site and your users from attack.
- Mixed content degrades the security and user experience of your HTTPS site.

Resource requests and web browsers

When a browser *visits* a website page, it is requesting for an HTML resource. The web server then returns the HTML content, which the browser parses and displays to users. Often a single HTML file isn't enough to display a complete page, so the HTML file includes references to other resources that the browser needs to request. These subresources can be things like images, videos, extra HTML, CSS, or JavaScript, which are each fetched using separate requests.

HTTPS benefits

When a browser requests resources over HTTPS—which stands for HTTP Secure—it uses an encrypted connection to communicate with the web server.

Using HTTPS has three main benefits:

- Authentication
- Data integrity

- Secrecy

Authentication

Is the website I'm talking to who they claim to be?

HTTPS lets the browser check that it has opened the correct website and hasn't been redirected to a malicious site. When navigating to your bank's website, your browser *authenticates* the website, thus preventing an attacker from impersonating your bank and stealing your login credentials.

Data integrity

Has anyone tampered with the content that I'm sending or receiving?

HTTPS lets the browser detect if an attacker has changed any data the browser receives. When transferring money using your bank's website, this prevents an attacker from changing the destination account number while your request is in transit.

Secrecy

Can anyone see the content I am sending or receiving?

HTTPS prevents an attacker from eavesdropping on the browser's requests, tracking the websites visited, or stealing information sent or received.

HTTPS, TLS, and SSL

HTTPS stands for HTTP Secure, Hyper(t)ext Transfer Protocol Secure. The **secure** portion here comes from the encryption added to the requests sent and received by the browser. Currently, most browsers use the TLS protocol to provide encryption; **TLS** is sometimes referred to as SSL.

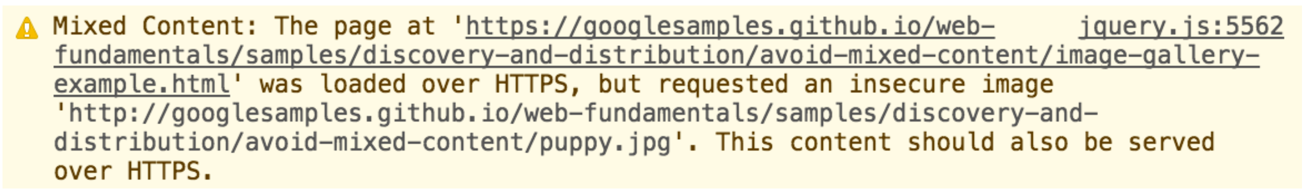
Details of HTTPS, TLS, and SSL are beyond the scope of this article, but if you want to learn more, the following resources are a good place to start:

- Wikipedia HTTPS
- Wikipedia TLS
- Khan Academy Cryptography course
- TLS chapter in High Performance Browser Networking by Ilya Grigorik

Mixed content weakens HTTPS

Requesting subresources using the insecure HTTP protocol weakens the security of the entire page, as these requests are vulnerable to **man-in-the-middle attacks**, where an attacker eavesdrops on a network connection and views or modifies the communication between two parties. Using these resources, an attacker can often take complete control over the page, not just the compromised resource.

Although many browsers report mixed content warnings to the user, by the time this happens, it is too late: the insecure requests have already been performed and the security of the page is compromised. This scenario is, unfortunately, quite common on the web, which is why browsers can't just block all mixed requests without restricting the functionality of many sites.



```
⚠ Mixed Content: The page at 'https://googlesamples.github.io/web-      jquery.js:5562
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/image-gallery-
example.html' was loaded over HTTPS, but requested an insecure image
'http://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/puppy.jpg'. This content should also be served
over HTTPS.
```

It's up to you, the developer, to fix mixed content issues in your application.

A simple example

Loading an insecure script from an HTTPS page.

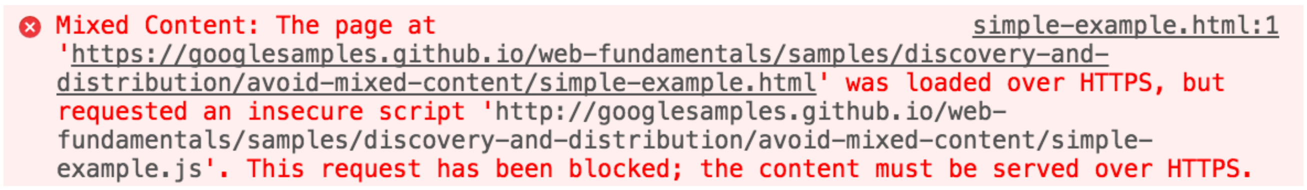
Viewing this sample page over **HTTPS**—https://googlesamples.github.io/web-fundamentals/.../simple-example.html—includes an **HTTP** script tag which attempts to load mixed content.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Materia
    <link rel="stylesheet" href="https://code.getmdl.io/1.2.1/material.indigo-pin
    <script defer src="https://code.getmdl.io/1.2.1/material.min.js"></script>
    <style>
      body {
        margin: 2em;
      }
    </style>
    <title>Simple mixed content example</title>
  </head>
  <body>
    <div role="main">
      <h1>
        Simple mixed content example!
      </h1>
      <p>
        View page over: <a href="https://googlesamples.github.io/web-fundamentals
      </p>
      <p>
        This page loads the script simple-example.js using HTTP. This is the simp
      </p>
      <div id="output">Waiting for insecure script to run...</div>
      <script src="https://googlesamples.github.io/web-fundamentals/samples/disco
    </div>
    <script>
  (function(b,o,i,l,e,r){b.GoogleAnalyticsObject=l;b[l]||(b[l]=
  function(){(b[l].q=b[l].q||[]).push(arguments)});b[l].l=+new Date;
  e=o.createElement(i);r=o.getElementsByTagName(i)[0];
  e.src='https://www.google-analytics.com/analytics.js';
  r.parentNode.insertBefore(e,r)}(window,document,'script','ga'));
  ga('create','UA-52746336-1');ga('send','pageview');
  var isCompleted = {};
  function sampleCompleted(sampleName){
    if (ga && !isCompleted.hasOwnProperty(sampleName)) {
      ga('send', 'event', 'WebCentralSample', sampleName, 'completed');
      isCompleted[sampleName] = true;
    }
  }
</script>
  </body>
</html>
```

Try it

In this example, the script **`simple-example.js`** is loaded with an **HTTP** URL. This is the simplest case of mixed content. When the browser requests the **`simple-example.js`** file, an attacker can inject code into the returned content and take control of the entire page.

Thankfully, most modern browsers block this type of dangerous content by default. See browser behavior with mixed content.



```
Mixed Content: The page at                            simple-example.html:1
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/simple-example.html' was loaded over HTTPS, but
requested an insecure script 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/simple-
example.js'. This request has been blocked; the content must be served over HTTPS.
```

Chrome blocks the insecure script.

An XMLHttpRequest example

Loading insecure data with XMLHttpRequest.

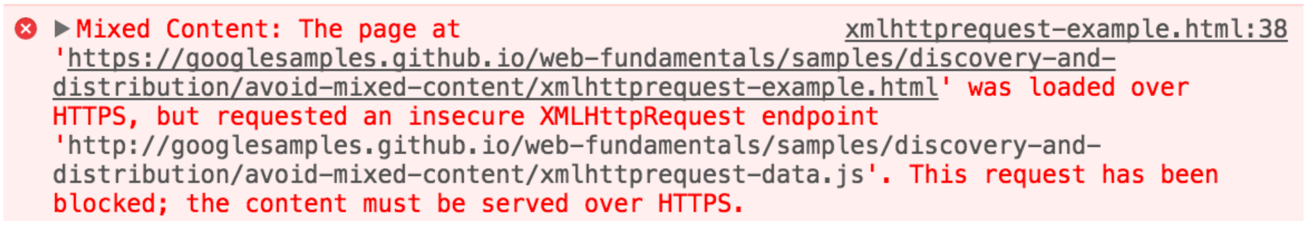
Viewing this sample page over **HTTPS**—**https**://googlesamples.github.io/web-fundamentals/.../xmlhttprequest-example.html—includes an `XMLHttpRequest` over **HTTP** to fetch mixed content `JSON` data.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Materia
    <link rel="stylesheet" href="https://code.getmdl.io/1.2.1/material.indigo-pin
    <script defer src="https://code.getmdl.io/1.2.1/material.min.js"></script>
    <style>
      body {
        margin: 2em;
      }
    </style>
    <title>XMLHttpRequest mixed content example</title>
  </head>
  <body>
    <div role="main">
      <h1>
        XMLHttpRequest mixed content example!
      </h1>
      <p>
        View page over: <a href="http://googlesamples.github.io/web-fundamentals/
      </p>
      <p>
        This page constructs an HTTP URL dynamically in JavaScript, the URL is ev
      </p>
      <div id="output">Waiting for data...</div>
      <script>
        var rootUrl = 'http://googlesamples.github.io/web-fundamentals/samples/di
        var resources = {
          jsonData: '/xmlhttprequest-data.js'
        };
        var request = new XMLHttpRequest();
        request.addEventListener('load', function() {
          var jsonData = JSON.parse(request.responseText);
          document.getElementById('output').innerHTML += '<br>' + jsonData.data;
        });
        request.open('GET', rootUrl + resources.jsonData, true);
        request.send();
      </script>
    </div>
    <script>
  (function(b,o,i,l,e,r){b.GoogleAnalyticsObject=l;b[l]||(b[l]=
  function(){(b[l].q=b[l].q||[]).push(arguments)});b[l].l=+new Date;
  e=o.createElement(i);r=o.getElementsByTagName(i)[0];
  e.src='//www.google-analytics.com/analytics.js';
  r.parentNode.insertBefore(e,r)}(window,document,'script','ga'));
  ga('create','UA-52746336-1');ga('send','pageview');
  var isCompleted = {};
  function sampleCompleted(sampleName){
    if (ga && !isCompleted.hasOwnProperty(sampleName)) {
      ga('send', 'event', 'WebCentralSample', sampleName, 'completed');
      isCompleted[sampleName] = true;
    }
  }
</script>
  </body>
</html>
```

Try it

Here the **HTTP** URL is constructed dynamically in JavaScript, and is eventually used by **`XMLHttpRequest`** to load an insecure resource. Like the simple example above, when the browser requests the **`xmlhttprequest-data.js`** file, an attacker can inject code into the returned content and take control of the entire page.

Most modern browsers block these dangerous requests as well.



```
▶ Mixed Content: The page at                    xmlhttprequest-example.html:38
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/xmlhttprequest-example.html' was loaded over
HTTPS, but requested an insecure XMLHttpRequest endpoint
'http://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/xmlhttprequest-data.js'. This request has been
blocked; the content must be served over HTTPS.
```

Chrome blocks the insecure XMLHttpRequest.

An image gallery example

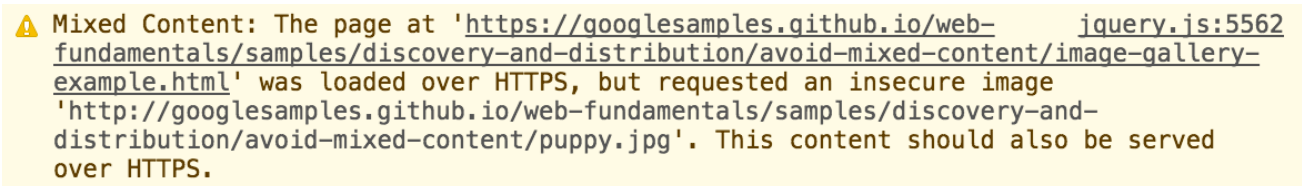
Loading insecure images with jQuery lightbox.

When viewing this sample page over **HTTPS**—**https**://googlesamples.github.io/web-fundamentals/.../image-gallery-example.html—initially it does not have any mixed content problems; however, when the thumbnail image is clicked, a full size mixed content image is loaded over **HTTP**.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Materia
    <link rel="stylesheet" href="https://code.getmdl.io/1.2.1/material.indigo-pin
    <script defer src="https://code.getmdl.io/1.2.1/material.min.js"></script>
    <style>
      body {
        margin: 2em;
      }
    </style>
    <title>Image gallery mixed content example</title>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></sc
    <script>
      $(document).ready(function() {
        $('.gallery').click(function(e) {
          e.preventDefault();
          $('.overlay-foreground').css('background-image', 'url(' + $(this).attr(
          $('.overlay').fadeIn('slow');
        })
        $('.overlay').click(function() {
          $('.overlay').fadeOut('slow');
        })
      });
    </script>
    <style>
      .overlay {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
      }
      .overlay-background {
        background-color: #000;
        filter:alpha(opacity=80);
        -moz-opacity: 0.8;
        -khtml-opacity: 0.8;
        opacity: 0.8;
        z-index: 10000;
      }
      .overlay-foreground {
        background-position: center center;
        background-repeat: no-repeat;
        z-index: 10001;
      }
    </style>
  </head>
  <body>
    <div role="main">
      <h1>
        Image gallery mixed content!
      </h1>
      <p>
        View page over: <a href="http://googlesamples.github.io/web-fundamentals/
      </p>
      <p>
        Image galleries often rely on the &lt;img&gt; tag src attribute to displa
      </p>
      CLICK ME! -->
      <a class="gallery" href="http://googlesamples.github.io/web-fundamentals/sa
        <img src="https://googlesamples.github.io/web-fundamentals/samples/discov
      </a>
      <div class="overlay overlay-background" style="display: none;"></div>
      <div class="overlay overlay-foreground" style="display: none;"></div>
    </div>
    <script>
  (function(b,o,i,l,e,r){b.GoogleAnalyticsObject=l;b[l]||(b[l]=
  function(){(b[l].q=b[l].q||[]).push(arguments)});b[l].l=+new Date;
  e=o.createElement(i);r=o.getElementsByTagName(i)[0];
  e.src='//www.google-analytics.com/analytics.js';
  r.parentNode.insertBefore(e,r)}(window,document,'script','ga'));
  ga('create','UA-52746336-1');ga('send','pageview');
  var isCompleted = {};
  function sampleCompleted(sampleName){
    if (ga && !isCompleted.hasOwnProperty(sampleName)) {
      ga('send', 'event', 'WebCentralSample', sampleName, 'completed');
      isCompleted[sampleName] = true;
    }
  }
</script>
  </body>
</html>
```

Try it

Image galleries often rely on the `<img>` tag `src` attribute to display thumbnail images on the page, the anchor (`<a>`) tag `href` attribute is then used to load the full sized image for the gallery overlay. Normally `<a>` tags do not cause mixed content, but in this case, the jQuery code overrides the default link behavior—to navigate to a new page—and instead loads the **HTTP** image on this page.



```
Mixed Content: The page at 'https://googlesamples.github.io/web-      jquery.js:5562
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/image-gallery-
example.html' was loaded over HTTPS, but requested an insecure image
'http://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/puppy.jpg'. This content should also be served
over HTTPS.
```

Insecure images degrade the security of your site, but they are not as dangerous as other types of mixed content. Modern browsers still load mixed content images, but display warnings to the user as well.

Mixed content types & security threats associated

The two types of mixed content are: active and passive.

Passive mixed content refers to content that doesn't interact with the rest of the page, and thus a man-in-the-middle attack is restricted to what they can do if they intercept or change that content. Passive mixed content includes images, video, and audio content, along with other resources that cannot interact with the rest of the page.

Active mixed content interacts with the page as a whole and allows an attacker to do almost anything with the page. Active mixed content includes scripts, stylesheets, iframes, flash resources, and other code that the browser can download and execute.

Passive mixed content

Passive mixed content still poses a security threat to your site and your users. For example, an attacker can intercept HTTP requests for images on your site and swap or replace these images; the attacker can swap the *save* and *delete* button images, causing your users to delete content without intending to; replace your product diagrams with lewd or pornographic content, defacing your site; or replace your product pictures with ads for a different site or product.

Even if the attacker doesn't alter the content of your site, you still have a large privacy issue where an attacker can track users using mixed content requests. The attacker can tell which pages a user visits and which products they view based on images or other resources that the browser loads.

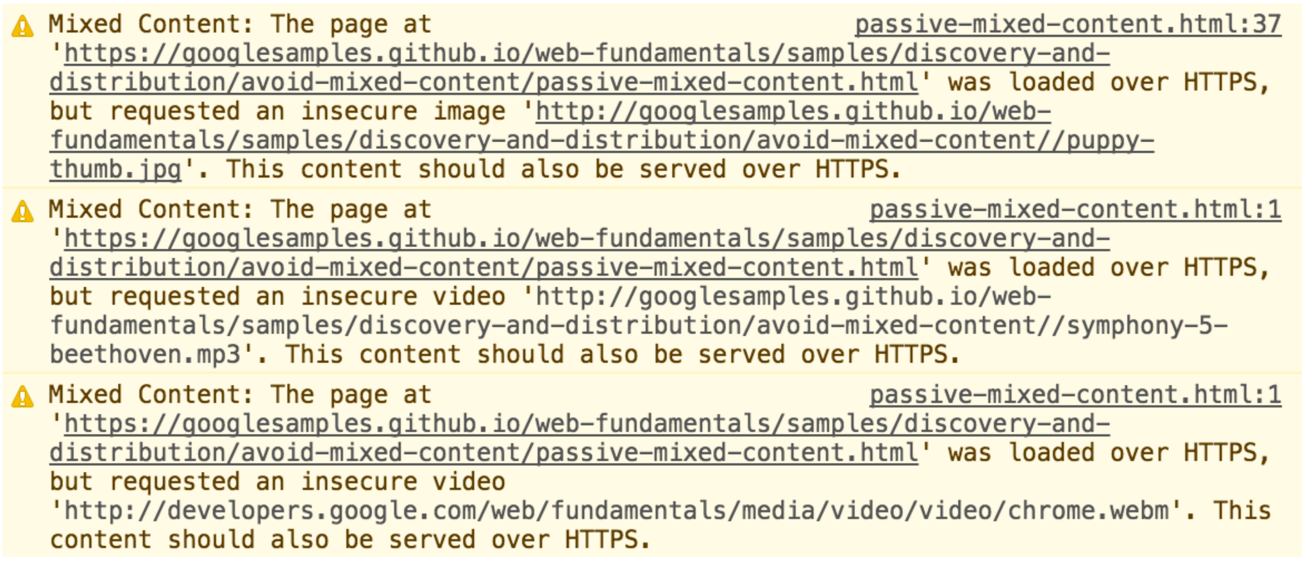
The following is an example of passive mixed content:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Materia
    <link rel="stylesheet" href="https://code.getmdl.io/1.2.1/material.indigo-pin
    <script defer src="https://code.getmdl.io/1.2.1/material.min.js"></script>
    <style>
      body {
        margin: 2em;
      }
    </style>
    <title>Passive mixed content example</title>
    <style>
      audio, img, video {
        display: block;
        margin: 10px;
      }
    </style>
  </head>
  <body>
    <div role="main">
      <h1>
        Passive mixed content!
      </h1>
      <p>
        View page over: <a href="http://googlesamples.github.io/web-fundamentals/
      </p>
      <p>
        Several examples of passive mixed content. When viewed over HTTPS most br
      </p>

      <!-- An insecure audio file loaded on a secure page -->
      <audio src="http://googlesamples.github.io/web-fundamentals/samples/discove

      <!-- An insecure image loaded on a secure page -->
      <img src="http://googlesamples.github.io/web-fundamentals/samples/discovery

      <!-- An insecure video file loaded on a secure page -->
      <video src="http://storage.googleapis.com/webfundamentals-assets/videos/chr
    </div>
    <script>
  (function(b,o,i,l,e,r){b.GoogleAnalyticsObject=l;b[l]||(b[l]=
  function(){(b[l].q=b[l].q||[]).push(arguments)});b[l].l=+new Date;
  e=o.createElement(i);r=o.getElementsByTagName(i)[0];
  e.src='//www.google-analytics.com/analytics.js';
  r.parentNode.insertBefore(e,r)}(window,document,'script','ga'));
  ga('create','UA-52746336-1');ga('send','pageview');
  var isCompleted = {};
```

```
  function sampleCompleted(sampleName){
    if (ga && !isCompleted.hasOwnProperty(sampleName)) {
      ga('send', 'event', 'WebCentralSample', sampleName, 'completed');
      isCompleted[sampleName] = true;
    }
  }
</script>
  </body>
</html>
```

Try it

Most browsers still render this type of mixed content to the user, however a warning is also displayed as this poses a security and privacy risk to your site and users.



```
⚠ Mixed Content: The page at                          passive-mixed-content.html:37
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/passive-mixed-content.html' was loaded over HTTPS,
but requested an insecure image 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content//puppy-
thumb.jpg'. This content should also be served over HTTPS.
⚠ Mixed Content: The page at                           passive-mixed-content.html:1
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/passive-mixed-content.html' was loaded over HTTPS,
but requested an insecure video 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content//symphony-5-
beethoven.mp3'. This content should also be served over HTTPS.
⚠ Mixed Content: The page at                           passive-mixed-content.html:1
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/passive-mixed-content.html' was loaded over HTTPS,
but requested an insecure video
'http://developers.google.com/web/fundamentals/media/video/video/chrome.webm'. This
content should also be served over HTTPS.
```

Mixed content warnings from the Chrome JavaScript console.

Active mixed content

Active mixed content poses a greater threat than passive. An attacker can intercept and rewrite active content, thereby taking full control of your page or even your entire website. This allows the attacker to change anything about the page, including displaying entirely different content, stealing user passwords or other login credentials, stealing user session cookies, or redirecting the user to a different site entirely.

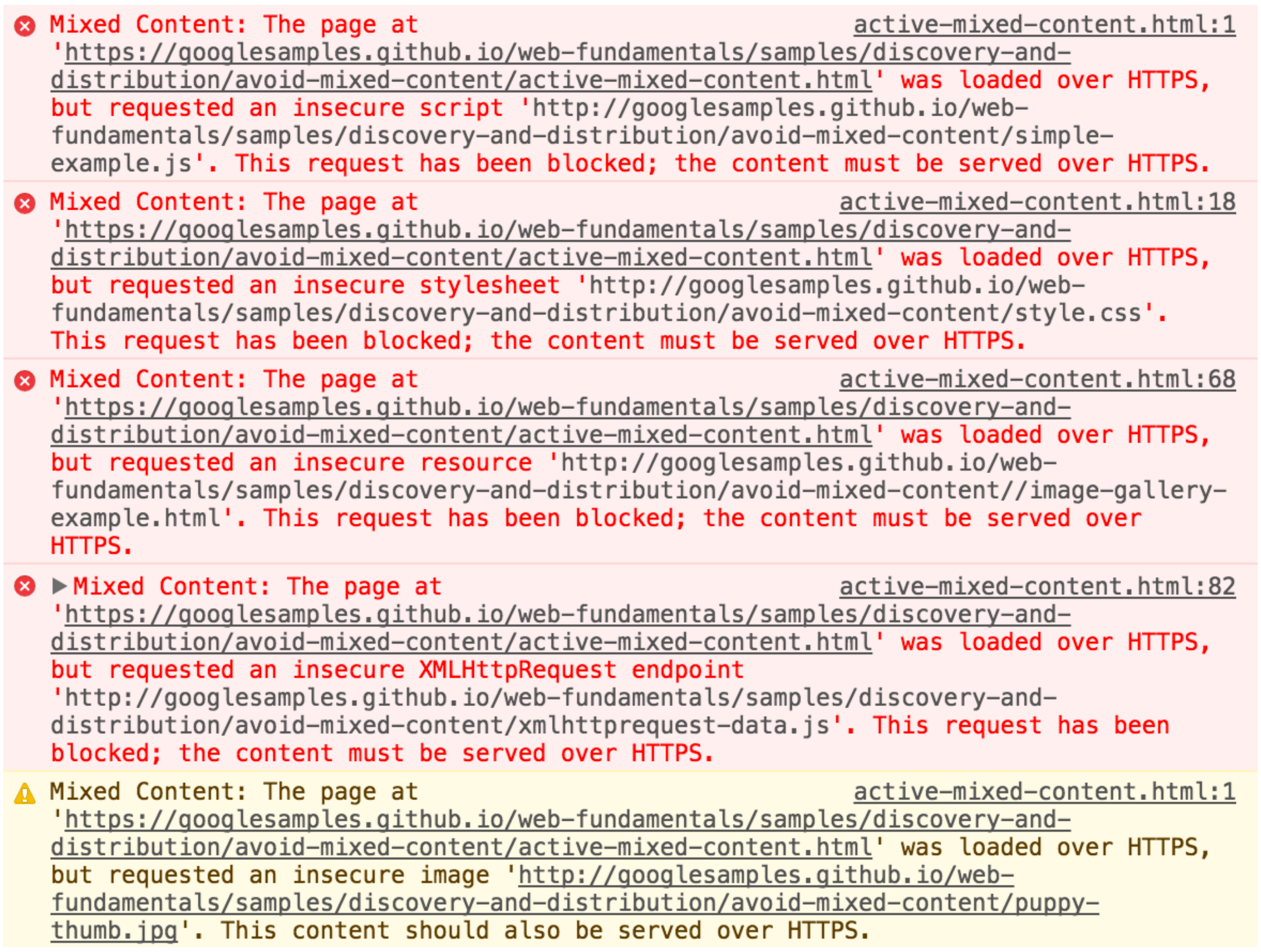
Due to the severity of this threat, many browsers block this type of content by default to protect users, but functionality varies between browser vendors and versions.

The following contains examples of active mixed content:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Materia
    <link rel="stylesheet" href="https://code.getmdl.io/1.2.1/material.indigo-pin
    <script defer src="https://code.getmdl.io/1.2.1/material.min.js"></script>
    <style>
      body {
        margin: 2em;
      }
    </style>
    <title>Active mixed content example</title>
    <!-- An insecure script file loaded on a secure page -->
    <script src="http://googlesamples.github.io/web-fundamentals/samples/discover

    <!-- An insecure stylesheet loaded on a secure page -->
    <link href="http://googlesamples.github.io/web-fundamentals/samples/discovery

    <style>
      .insecure-background {
        /* An insecure resources loaded from a style property on a secure page, t
           happen in many places including, @font-face, cursor, background-image,
        background: url('http://googlesamples.github.io/web-fundamentals/samples/
      }
    </style>

    <style>
      .insecure-style-holder span {
        color: #fff;
      }
      .insecure-background {
        color: #000;
        font-weight: bold;
        background-position: left center;
        background-repeat: no-repeat;
        width: 300px;
        height: 140px;
      }
      iframe {
        width: 400px;
        height: 300px;
      }
    </style>

  </head>
```

```
<body>
  <div role="main">
    <h1>
      Active mixed content!
    </h1>
    <p>
      View page over: <a href="http://googlesamples.github.io/web-fundamentals/
    </p>
    <p>
      Several examples of active mixed content. When viewed over HTTPS most bro
    </p>
    <div class="insecure-style-holder">
      <span style="ba">Insecure style loaded</span>
    </div>
    <div class="insecure-background">
      Loading insecure background here...
    </div>

    <p>Loading insecure iframe...</p>
    <!-- An insecure iframed page loaded on a secure page -->
    <iframe src="http://googlesamples.github.io/web-fundamentals/samples/discov

    <!-- Flash resources also qualify as active mixed content and pose a
    serious security risk. Be sure to look for <object> tags with type set
    to "application/x-shockwave-flash", and an http:// data attribute. -->
    <!-- <object type="application/x-shockwave-flash" data="http://..."></objec

    <script>
      // An insecure resource loaded using XMLHttpRequest
      var request = new XMLHttpRequest();
      request.addEventListener('load', function() {
        var jsonData = JSON.parse(request.responseText);
        document.getElementById('output').innerHTML += '<br>' + jsonData.data;
      });
      request.open("GET", "http://googlesamples.github.io/web-fundamentals/samp
      request.send();
    </script>
    <div id="output">Waiting for insecure script to run...</div>
  </div>
  <script>
(function(b,o,i,l,e,r){b.GoogleAnalyticsObject=l;b[l]||(b[l]=
function(){(b[l].q=b[l].q||[]).push(arguments)});b[l].l=+new Date;
e=o.createElement(i);r=o.getElementsByTagName(i)[0];
e.src='//www.google-analytics.com/analytics.js';
r.parentNode.insertBefore(e,r)}(window,document,'script','ga'));
ga('create','UA-52746336-1');ga('send','pageview');
var isCompleted = {};
function sampleCompleted(sampleName){
    if (ga && !isCompleted.hasOwnProperty(sampleName)) {
      ga('send', 'event', 'WebCentralSample', sampleName, 'completed');
      isCompleted[sampleName] = true;
    }
  }
</script>
  </body>
</html>
```

Try it



```
Mixed Content: The page at                              active-mixed-content.html:1
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/active-mixed-content.html' was loaded over HTTPS,
but requested an insecure script 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/simple-
example.js'. This request has been blocked; the content must be served over HTTPS.
Mixed Content: The page at                             active-mixed-content.html:18
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/active-mixed-content.html' was loaded over HTTPS,
but requested an insecure stylesheet 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/style.css'.
This request has been blocked; the content must be served over HTTPS.
Mixed Content: The page at                             active-mixed-content.html:68
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/active-mixed-content.html' was loaded over HTTPS,
but requested an insecure resource 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content//image-gallery-
example.html'. This request has been blocked; the content must be served over
HTTPS.
▶ Mixed Content: The page at                           active-mixed-content.html:82
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/active-mixed-content.html' was loaded over HTTPS,
but requested an insecure XMLHttpRequest endpoint
'http://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/xmlhttprequest-data.js'. This request has been
blocked; the content must be served over HTTPS.
Mixed Content: The page at                              active-mixed-content.html:1
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/active-mixed-content.html' was loaded over HTTPS,
but requested an insecure image 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/puppy-
thumb.jpg'. This content should also be served over HTTPS.
```

Mixed content errors from the Chrome JavaScript console.

Browser behavior with mixed content

Due to the threats described above, it would be ideal for browsers to block all mixed content. However, this would break a large number of websites that millions of users rely on every day. The current compromise is to block the most dangerous types of mixed content and allow the less dangerous types to still be requested.

Modern browsers follow mixed content specification, which defines **optionally blockable content** and **blockable content** categories.

From the spec, a resource qualifies as optionally blockable content "when the risk of allowing its usage as mixed content is outweighed by the risk of breaking significant portions of the web"; this is a subset of the passive mixed content category described above. At the time of this writing, images, video, and audio resources, as well as prefetched links, are the only resource types included in optionally blockable content. This category is likely to get smaller as time goes on.

All content that is not **optionally blockable** is considered **blockable**, and is blocked by the browser.

Browser versions

It is important to remember that not every visitor to your website use the most up-to-date browsers. Different versions from different browser vendors each behave differently with mixed content. At worst, some browsers and versions don't block any mixed content at all, which is very unsafe for the user.

The exact behavior of each browser is constantly changing, so we won't include specifics here. If you're interested in how a specific browser behaves, look for information published by the vendors directly.

Note: Your users are counting on you to protect them when they visit your website. It is important to fix your mixed content issues to protect **all** your visitors, including those on older browsers.

Next
Preventing Mixed Content →