HTTPS 安全最佳实践（二）之安全加固

1017464136725776 / 2018-03-07 15:32:00 / 浏览数 10626 技术文章 技术文章 顶(0) 踩(0)

Author:HENRY.CHEN
https://blog.myssl.com/https-security-best-practices/

当你的网站上了 HTTPS 以后，可否觉得网站已经安全了？这里 提供了一个 HTTPS 是否安全的检测工具，你可以试试。

本篇正文讲述的是 HTTP 安全的最佳实践，着重在于 HTTPS 网站的 Header 的相关配置。

1 连接安全性和加密

1.1 SSL/TLS

传输层安全（TLS）及其前身安全套接字层（SSL），通过在浏览器和 web 服务器之间提供端到端加密来促进机密通信。没有 TLS，就谈不上什么安全。TLS 是 HTTP 安全性的基础。

想要部署 TLS 是非常容易的，但其难点在于如何使用安全的配置来保障站点的安全。
尤其是 Protocol 版本和 Cipher 需要小心选择和配置。你可以通过本站 工具 体检你的网站，发现并解决这些细节的问题。

建议
所有本地和链接的资源需要正确的配置，且要使用 TLS。

1.2 HTTP Strict Transport Security (HSTS)

指示浏览器只使用 HTTPS 连接到目标服务器。这可以防止一些潜在的中间人攻击，包括 SSL 剥离，会话 cookie 窃取（如果没有被适当保护）。如果遇到任何与证书相关的错误，它还可以阻止浏览器连接到网站。当浏览器访问一个设置相应 HTTP header 的 HTTPS 网站时，HSTS 将被激活。

HSTS 有一个固定期限，由 max-age 字段值控制。这个值可以是静态的，也可以是相对于将来某个特定日期的，你可以设置成 SSL 证书的过期时间。

在浏览器中，HSTS 首选项可以通过提交到 Chromium's HSTS preload list 来硬编码，这是所有实现 HSTS 使用的浏览器。

注意，HSTS 确实有陷阱。它提供了 `include subdomains` 选项，这在实践中可能是太宽泛了。此外，客户端错误可能会造成严重的后果——客户端错误的时钟导致它认为服务器的 SSL 证书无效或过期，或者缺少根 CA 证书——将不再导致浏览器中的证书错误。浏览器将完全拒绝访问页面，并且可能会显示让安全专家之外的完全无法理解的错误。

建议
设置 HSTS header 长的生命周期，最好是半年及以上。

```
Strict-Transport-Security: max-age=31536000
```

1.3 Public Key Pins

HTTP PKP（HPKP）指示浏览器只与提供的 SSL/TLS 的 HASH 相符或存在于同一证书链的服务器相连接。换句话说，如果 SSL/TLS 证书以一种意想不到的方式发生了变化，浏览器就无法连接到主机。这主要是针对受信任证书颁发机构（CA）或流氓 CA 证书颁发的伪造证书，用户可能会被骗安装。

例如，浏览器连接到 `https://example.com`，它存在这个头。header 告诉浏览器，如果证书 key 匹配，或者在发出证书链中有一个 key 匹配，那么在将来才会再次连接。其他的指令组合是可能的。它们都极大地减少了攻击者在客户端和合法主机之间模拟主机或拦截通信的可能性。

像 HSTS 一样，HPKP 在实现之前需要仔细的思考和计划。错误可以将用户锁定在您的站点之外，并且不容易修复。

像 HSTS 一样，HPKP 在实现之前需要仔细的思考和计划。错误可以将用户锁定在您的站点之外，并且不容易修复。

建议
确定是否需要为您的站点使用 PKP。如果是这样的话，那么从一个较小的实践开始，如果在一段时间之后没有遇到问题，就增加它。如果 SSL/TLS 密钥需要更新，建立备份计划。优先创建备份密钥和离线存储。

示例HTTP头:

```
Public-Key-Pins: max-age=5184000; pin-sha256="+oZq/vo3Kcv0CQPjpdwyInqVXmLiobmUJ3FaDpD/U6c="; pin-sha256="47DEQpj8HBSa+/TImW+5J
```

1.4 Mixed HTTPS and HTTP Content

主站点通过 HTTPS 安全地服务，但是在 HTTP 上加载一些文件（images、js、css）。这是一个巨大的安全漏洞，破坏了 HTTPS 提供的安全性。受影响的站点可能会泄漏会话 cookie 或用户行为信息。它们也可能容易受到注入和其他 MITM 攻击的攻击，而 HTTPS 通常会阻止这种攻击。

建议
如果 HTTPS 部署在主站上，请将任何地方的所有内容都 HTTPS 化（全站 HTTPS）。

2 Content security

2.1 Content Security Policy

为浏览器提供关于网站内容类型和行为的明确说明。良好的内容安全策略（CSP）可以帮助抵御跨站点脚本（XSS）和其他注入攻击等攻击。CSP 支持所有主要的浏览器，尽管只是部分地之前在 IE 11。

一个好的 CSP 是基于白名单的方法，不允许任何东西，除了明确允许的内容。它还限制了 javascript 的来源和允许操作。

CSP 很难启用遗留代码库。为了简化实现，CSP 提供了一个 `report-only` 模式，在浏览器中，CSP 的违规被发送到一个网站端点，但是该策略没有被强制执行。
新项目应该从一开始就使用 CSP。

建议
从限制性政策开始，在必要时放松。禁止所有的例子:

```
Content-Security-Policy: default-src 'none';
```

现在让我们允许自托管 scripts、images、CSS、fonts 和 AJAX，以及 jQuery CDN 托管脚本和 Google Analytics:

```
Content-Security-Policy: default-src 'none'; script-src 'self' https://code.jquery.com https://www.google-analytics.com; img-s
```

要注意的是，不要让所有的东西都破坏你的网站，例如，如果你使用 `child-src` 指令，而浏览器不支持它。一个不那么严格的政策可能从以下开始:

```
Content-Security-Policy: default-src 'self';
```

甚至更少的限制性政策甚至可以使用 `default-src '*'`，然后添加限制。我建议你不要这么做，除非你完全明白其中的含义。否则，你可能会依赖 CSP，它只会给你一种错误的安全感。

2.2 Frame Options

控制站点是否可以放置在 `<iframe>`，`<frame>` 或 `<object>` 标签。不允许使用框架可以防止 clickjacking 攻击。例如，从 2015 年 2 月起，Internet Explorer's universal cross-site-scripting bug 就被这个消息头减轻了。

`X-Frame-Options` 是一个非标准的 header，在内容安全策略级别 2 中被 frame ancestor指令所取代。然而，frame ancestor 还没有得到普遍的支持，而 `X-Frame-Options` 得到了广泛的支持。

建议
确定你的网站是否需要被允许呈现在一个 frame 中。完全不允许使用 sameorigin 拒绝或允许同源框架的选项。避免由于受限或 bug 浏览器支持而允许的选项。示例 HTTP 头:

```
X-Frame-Options: deny
```

2.3 XSS Protection

跨站点脚本（XSS 或 CSS）的保护被构建到大多数流行的浏览器中，除了 Firefox 之外。这种保护是用户可配置的，可以关闭。因此，明确要求浏览器在你的网站上使用它的 XSS 过滤器是个好主意。

相反，网站可以要求 XSS 保护在页面的基础上被禁用。这绝对不是一个好主意。

建议
使用入校 HTTP header:

```
X-Xss-Protection: 1; block
```

2.4 Cache Control

表示缓存页面输出的首选项。适当的值随网站数据的性质而变化，但强烈推荐使用偏好。否则，它取决于浏览器和代理来选择是否缓存内容。不恰当的选择可能会导致性能问

建议
开发缓存策略，然后将缓存首选项包括为 HTTP 头。

```
Cache-Control: public*
```

其中的一个 public，private，no-cache 或 no-store。如果允许缓存，则应该将 max-age 值包含在 Cache-Control 以及 Etag 头文件中，以允许客户端缓存验证。

2.5 Content Type Options

当浏览器以不同的方式处理来自服务器的文件时，MIME 嗅探就是服务器指令。当一个网站承载不受信任的内容（如用户提供的）时，这是很危险的。假设服务器允许用户上传 image。如果用户上传 HTML 文档，浏览器可能会将其呈现为 web 执行 scriptpage，即使服务器明确表示正在发送 image。非标准的标头 `X-Content-Type-Options`选项指示浏览器不做任何模仿指定类型的 MIME。

建议
总是设置 header:

```
X-Content-Type-Options: nosniff
```

2.6 Subresource Integrity

浏览器通常从外部域加载大量资源、javascript
和样式表。内容交付网络经常被使用。如果外部资源被破坏，依赖站点的安全性也可以。子资源完整性允许浏览器验证 javascript 或样式表未被意外修改。

建议
设置外部 javascript 和样式表的完整性属性。

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js" integrity="sha384-6ePHh72Rl3hKio4HiJ841psfsRJv
```

注意
您应该始终提供外部脚本的本地副本，并实现一种方法，以便在外部负载失败的情况下重新加载它们。否则你的网站可能会崩溃。例子:

```
<script>window.jQuery || document.write('<script src="/jquery.min.js"><\/script>')</script>
```

2.7 Iframe Sandbox

iframe 在 WWW 上随处可见。网站平均有 5.1 iframe，主要用于装载第三方内容。这些 iframe
有很多方法来伤害托管网站，包括运行脚本和插件和重新引导访问者。`sandbox`属性允许对 iframe 中可以进行的操作进行限制。

建议
设置 iframe 的 `sandbox` 属性，然后添加所需的权限。

```
<iframe src="https://example.com" sandbox="allow-same-origin allow-scripts"></script>
```

2.8 Server Clock

服务器包括所有响应的时间戳。不准确的时钟不会给客户机浏览器带来问题。然而，当与其他系统或服务交互时，问题就会出现。

建议
使用网络时间协议（NTP）来保持服务器时钟的准确性。

3 Information disclosure

3.1 Server Banner

大多数 web
服务器设置报头来识别自己和他们的版本号。这只服务于信息目的和实际用途是非常有限的。去掉整个头，而完全可以接受，通常是不必要的。但是，建议从头中删除版本号
web 服务器版本中存在 bug 的情况下，包括版本号可以作为对脚本 kiddy 的邀请来尝试对服务器的攻击。

建议
包含服务器名称但去掉版本号；

```
Server: nginx
```

3.2 Web Framework Information

许多 web 框架设置 HTTP
头，识别框架或版本号。除了满足用户的好奇心，而且主要作为技术堆栈的广告，这几乎没有什么作用。这些头是不标准的，对浏览器渲染站点的方式没有影响。

虽然它们没有什么实际用途，但对于搜索运行过时版本的软件的机器人或蜘蛛来说，这些标头是无价的，因为这些软件可能包含安全漏洞。如果没有定期更新，这些头文件可

建议
从服务器响应中删除这些标头: `X-Powered-By`, `X-Runtime`, `X-Version` 和 `X-AspNet-Version`。

4 Cookies

4.1 Cookie Security

包含敏感信息的 cookie，特别是会话 id，需要标记为安全的，假设网站是通过 HTTPS 传输的。这会阻止 cookie 通过 HTTP 发送明文文本。另一种方法是通过
HSTS 来阻止非安全 cookie 在 HTTP 上传输。建议使用安全 cookie 和 HSTS。

会话 cookie 应该与 HttpOnly 值进行标记，以防止它们被 javascript 访问。这可以防止攻击者利用 XSS 窃取会话 cookie。其他 cookie
可能不需要这样标记。但是，除非有明确的需要从 javascript 中访问他们的值，否则最好还是呆在安全的一边，把所有cookie标记为 HttpOnly

建议
标记所有 cookie 安全和 HttpOnly。

```
Set-Cookie: Key=Value; path=/; secure; HttpOnly, Key2=Value2; secure; HttpOnly
```

点击收藏 | 0 关注 | 3

上一篇：HTTPS 安全最佳实践（一）之S... 下一篇：【漏洞复现】Flash 0day漏...

1. 0 条回复
   - 动动手指，沙发就是你的了！