doi:10.11959/j.issn.1000-436x.2016206

HTML5 应用程序缓存中毒攻击研究

贾岩[1]，王鹤[1]，吕少卿[1]，张玉清[1,2]

（1. 西安电子科技大学综合业务网理论及关键技术国家重点实验室，陕西 西安 710071；
2. 中国科学院大学国家计算机网络入侵防范中心，北京 101408）

摘 要：HTML5 应用程序缓存使浏览器可以离线地访问 Web 应用，同时也产生了新的缓存中毒攻击手段。首先，对应用程序缓存中毒攻击的原理及危害进行了分析，然后针对使用应用程序缓存的站点，首次提出了 2 次替换 manifest 文件的新式缓存中毒攻击方法 RFTM。在 RFTM 攻击中，服务器端不会收到客户端发送的异常 HTTP 请求，故对服务器进行配置无法防范，攻击更具隐蔽性。最后设计了一套能有效防止此类攻击的应用层轻量级签名防御方案 Sec-Cache。实验表明 Sec-Cache 防御方案能够有效地防御 RFTM 攻击，并有良好的性能与兼容性。

关键词：Web 安全；HTML5；应用程序缓存；缓存中毒攻击；签名方案
中图分类号：TP393.08 **文献标识码**：A

Research on HTML5 application cache poison attack

JIA Yan[1], WANG He[1], LYU Shao-qing[1], ZHANG Yu-qing[1,2]

(1. Information Security Research Center of State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China;
2. National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 101408, China)

Abstract: HTML5 application cache (AppCache) allowed Web browser to access Web offline. But it also brought a new method of cache poisoning attack that was more persisting. As for websites which used the AppCache, a novel poisoning method RFTM (replace file twice method), in which the attacker replaced the manifest file twice to poison the client's AppCache, was proposed. Compared with the original attack, the legal server would not receive abnormal HTTP requests from the client in the attack. Therefore, changing the server configuration could not prevent the client from the RFTM AppCache poisoning. To avoid the attack mentioned above, a lightweight signature defense scheme Sec-Cache in application layer was designed. Furthermore, experiments show that it has good performance and compatibility.

Key words: Web security, HTML5, application cache, cache poisoning attack, signature scheme

1 引言

随着互联网的飞速发展，Web 已经不单是用来浏览简单的文档，而成为一个越来越丰富的平台。早期的 Web 标准 HTML4（超文本标记语言第 4 代，hyper text markup language 4）已经无法满足人们的需求，因此，W3C 组织于 2014 年 10 月正式发布 HTML5 标准。利用 HTML5 的新特性，开发者能够方便地在不同平台实现离线文档处理、地理信息查找和定位等功能。目前的主流浏览器如 IE、Firefox、Chrome 和 Opera 等，都对 HTML5 提供了良好的支持[1]。

HTML5 在提供各种便利的同时，也带来了新的安全问题。在 2010 年的信息安全行业的最高盛会——黑帽大会上，Kuppan 提出了多条 HTML5 存在的安全风险，其中，首次提出了应用程序缓存中毒攻击[2]。之后，针对 HTML5 新特性的安全问题，国内外研究人员都做了大量的研究，包括 Geolocation[3]、postMessage API[4]、跨域资源共享（CORS, cross- origin resource sharing）[5]、WebSocket[6]

收稿日期：2016-03-11；修回日期：2016-07-25
基金项目：国家自然科学基金资助项目（No.61272481, No.61572460）；教育部—中国移动科研基金资助项目（No.MCM20130431）
Foundation Items: The National Natural Science Foundation of China (No.61272481, No.61572460), Research Fund of Ministry of Education−China Mobile (No.MCM20130431)

以及移动平台 HTML5 App 的跨站脚本攻击（XSS, cross site scripting）[7]等。

其中，HTML5 引入的应用程序缓存（AppCache, applications cache）能够将网站的内容缓存在本地，使用户能够在离线的情况下继续访问，方便了离线使用并减少了网络流量，但同时也带来了新的缓存中毒安全威胁。Gilger[8]详细阐述了 Kuppan[2]提出的应用程序缓存中毒攻击，即通过中间人等手段，利用应用程序缓存更新机制的缺陷使用户缓存恶意内容，在受害者回到正常网络环境后，访问合法网站也会持续使用缓存的恶意内容。在这种攻击中，攻击者仅需通过中间人或 DNS 欺骗等手段在用户浏览器中缓存恶意代码一次，便可让用户之后每次访问该网站时都使用恶意内容，从而达到长期劫持受害者浏览器的目的，即使使用 HTTPS 也不能完全避免此类攻击[9]。但是，在这种应用程序缓存中毒攻击中，浏览器会向合法服务器端请求不存在的应用程序缓存清单（manifest）文件，并要求返回满足条件的 HTTP 响应（如返回 30X 的 HTTP 状态码），实现难度较大且会在服务器日志中留下异常记录。因此，对于这种攻击方式，在应用层通过服务器合理配置即可防范。

本文在现有 HTML5 应用程序缓存中毒攻击的研究基础之上，提出了一种 2 次替换 manifest 文件的新攻击方式，称为 2 次文件替换法（RFTM，replace file twice method）应用程序缓存中毒攻击。与 Kuppan 所提出的攻击相比，RFTM 攻击使客户端原 manifest 文件保持不变，合法服务器端将不会收到任何异常请求，从而绕过 Web 应用防火墙等检测工具，攻击更加隐蔽。针对 HTML5 应用程序缓存攻击以及 RFTM 攻击，本文从传输层和应用层这 2 个角度探讨了相应的防御方案，设计并实现了一套不需要数字证书认证（CA，certificate authority）中心的轻量级签名方案 Sec-Cache 来防止 manifest 文件被攻击者替换。

本文的贡献主要包括以下几点。

1) 提出了一种新的 2 次替换 manifest 文件法 HTML5 应用程序缓存中毒攻击方式。与传统的应用程序缓存中毒攻击相比，RFTM 攻击更加隐蔽。

2) 针对 RFTM 攻击，设计了一套防御方案 Sec-Cache。Sec-Cache 通过不需要数字证书认证中心的轻量级签名来防止 manifest 文件被攻击者替换，进而防止客户端的应用程序缓存被恶意污染。

3) 基于 PHP 脚本语言和 IE 浏览器插件实现了所提出的 Sec-Cache 防御方案，并进行了相应测试。实验表明本文提出的防御方案能够有效地防御 RFTM 攻击，并有良好的性能与兼容性。

2 相关工作

目前，HTML5 引入的许多安全问题得到了广泛关注，如文献[4,10,11]研究了 postMessage API 实现与使用方面的安全问题，文献[12,13]发现了多媒体类新特性产生的隐私泄露和 XSS 攻击等。

在众多新特性中，HTML5 应用程序缓存带来的安全问题也吸引了许多研究者的目光。Johns 等[14]发现使用 AppCache 缓存恶意脚本致 DNS-IP 映射信息过期，可以绕过反 DNS Rebinding 机制，从而破坏浏览器的同源策略。Lee 等[15]提出了利用应用程序缓存的事件机制来判断跨源资源状态的方法，从而推断出用户的访问习惯与认证状态。这些工作关注 DNS Rebinding 和用户状态信息泄露，而没有充分研究 AppCache 缓存中毒机制。Homakov[16]结合 HTTP 缓存和 AppCache 特性，为攻陷的站点建立长期后门，但该方法易受客户端刷新或清除缓存影响；而 RFTM 攻击完全符合 AppCache 的 W3C 标准，缓存中毒不易清除。Kuppan[2]和 Gilger[8]所述攻击方式利用客户端处理 manifest 文件的逻辑对任意网站进行缓存中毒攻击，但是服务端易觉察出流量异常，从而能够进行相应配置来防范该攻击；而 RFTM 攻击中没有异常流量，更加隐蔽。

3 背景介绍

3.1 HTML5 应用程序缓存

HTML5 引入了应用程序离线缓存[17]，浏览器每次只需从服务器下载更改过的资源，使 Web 应用可以在没有互联网连接时进行离线浏览，并且能够加快网站访问速度，同时减少服务器负载。

使用 HTML5 应用程序缓存，需在文档的<html>标签中包含 manifest 属性，例如<html manifest= “demo.appcache”>，其中，demo.appcache 就是应用程序缓存清单文件，称为 manifest 文件。manifest 文件是一个简单的文本文件，它告知浏览器需要被缓存的内容以及不要被缓存的内容，当前页面不在文件中指明也会被默认缓存。并且，manifest 文件需要在服务器上配置为正确的多用途互联网邮件扩展类型（MIME, multipurpose Internet

mail extensions），即 “text/cache-manifest”，其推荐的扩展名是 appcache。

如图 1 所示，一个完整的 manifest 文件需要以 CACHE MANIFEST 开头，接下来以#开头的行表示注释。在这个例子中，要缓存的文件是 logo.gif 和 main.js；需要每次连接请求的是 login.asp；而如果无法建立互联网连接，html5 文件夹下的文件会被 404.html 替代。

```
1   CACHE MANIFEST
2   # 2015-09-15 v1.0.0
3   /logo.gif
4   /main.js
5
6   NETWORK:
7   login.asp
8
9   FALLBACK:
10  /html5/ /404.html
```

图 1　manifest 文件示例

如图 2 所示，对于使用了 HTML5 应用程序缓存的页面，浏览器在第一次访问时，会下载网页的内容及 html 标签中指明的 manifest 文件，并根据文件中的内容缓存指定的文件。接下来再次对该 URL 访问时，浏览器会根据缓存内容首先请求下载 manifest 文件，若检验该文件没有更改，则直接使用缓存的内容，不再下载指定缓存的文件。即使服务器端更改了 main.js 的内容，若 manifest 文件没有更改，浏览器仍然会使用缓存的 main.js 文件。

浏览器处理应用程序缓存的逻辑如图 3 所示。如果浏览器已经缓存了合法站点的内容，用户再次访问该站点时，会根据之前记录的 URL 首先请求 manifest 文件，判断服务器返回的 HTTP 状态码。若服务器返回 404 或 410，应用程序缓存会被删除，浏览器重新请求资源；若返回其他错误代码或重定向，则应用程序缓存不会被删除；若服务器返回的是 200 状态码，浏览器接下来会判断 MIME 类型，若是要求的 text/cache-manifest MIME 类型，则进入对比 manifest 文件阶段，否则继续使用原来的缓存。在对比阶段，若发现 manifest 文件发生了更改，则更新 manifest 文件以及缓存的内容。

3.2　应用程序缓存中毒攻击

应用程序缓存中毒攻击是指通过应用程序缓存让受害者在访问合法网站时使用攻击者提供的恶意内容，而且攻击者加入的代码可以长期存在于受害者的浏览器中。HTTP 缓存和 HTML5 应用程序缓存都可以达到劫持用户浏览的目的[18]，但相比于 HTTP 缓存，HTML5 应用程序缓存提供了新的缓存毒化手段，且恶意缓存至少会使用一次，不会轻易受到刷新网页的影响，毒化的持续时间更长。

3.2.1　攻击条件

攻击者只要达到以下 2 个条件之一即可发动应用程序缓存中毒攻击。1) 攻击者能够控制受害者浏览器与合法域名的通信，如通过局域网 ARP 欺骗、钓鱼 AP[19]来进行中间人流量劫持，或使用 DNS 欺骗等手段。2) 攻击者攻陷合法站点，从而可以在站点中加入应用程序缓存以及恶意内容，使访问该站点的所有用户缓存恶意内容。只需满足以上任何一个条件，攻击者即可为合法域名加入自己编写的 manifest 文件和恶意脚本等内容。

3.2.2　攻击流程

要实现应用程序缓存攻击，攻击者需要让受害者浏览器在访问合法网站时使用加入的恶意内容。Kuppan 所提攻击就是根据图 3 中的①、②这 2 条路径使受害者长期使用缓存的恶意内容，直到用户手动删除应用程序缓存。其攻击主要包含以下几个步骤（如图 4 所示）。

1) 前期准备，如调查用户访问习惯、模仿站点、编写恶意 JavaScript 等。

2) 通过钓鱼 Wi-Fi、中间人、DNS 欺骗等方式对目标 URL 网页注入 manifest 属性，使用户浏览器下载配置正确 MIME 类型的 manifest 文件，从而缓存恶意内容。

3) 确保 manifest 文件的 URL 在合法网站不会

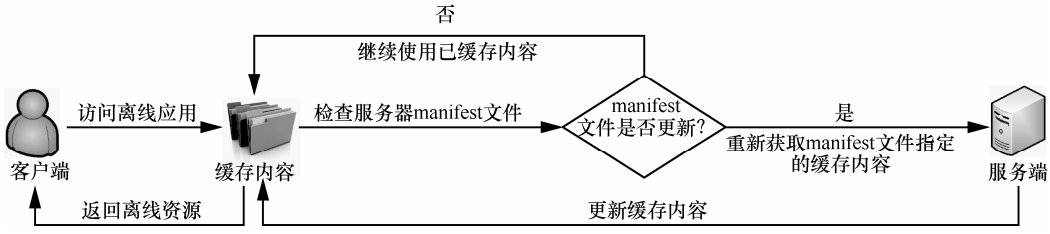

图 2　正常使用应用程序缓存的情况

返回 404 或者 410 状态码，或者返回含有非 text/cache-manifest 类型的 200 状态码。

4) 用户返回正常的网络环境访问网站，却使用了攻击者缓存的恶意内容，此时，攻击结束。

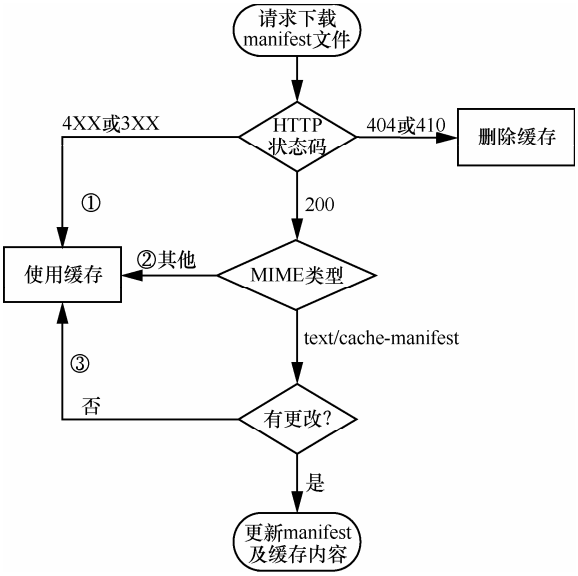

图 3 浏览器处理应用程序缓存逻辑

3.2.3 攻击危害

结合 HTML5 丰富的功能，攻击者可以利用应用程序缓存中毒攻击进行许多复杂的攻击。如 shell of the future[20]（一种反向 Web shell 控制器）即可以作为攻击手段来长期劫持用户的会话。除会话劫持外，攻击者利用 WebSocket 可以对受害者所在的局域网进行扫描[21]，找到内部服务器的 IP 地址，并利用 CORS 反馈给攻击者，泄露内部网络信息。利用应用程序缓存还可以方便攻击者判断受害者内部网络的 URL，甚至其登录状态[15]。若移动端 HTML5 App 缓存了恶意内容，攻击者利用 Geolocation API 还能够获知受害者的地理位置信息。

通过浏览器，HTML5 给予了攻击者强大的操作能力，甚至可以攀比传统 PC 端的本地木马，而 Web 的跨平台特性使攻击可以容易地扩展到移动端。应用程序缓存中毒攻击就是让受害者感染这种“木马”的手段。

4 RFTM 应用程序缓存中毒攻击

Kuppan 所描述的攻击方式采用图 3 中的第①、②条标号路径来达到让用户缓存恶意内容的效果，由于缓存中毒的客户端会尝试下载不存在的 manifest 文件，所以服务器会收到异常的 GET 请求，从而会触发服务端的防御措施，留下攻击痕迹。因此，本文提出了 2 次替换 manifest 文件的攻击方法 RFTM，通过图 3 中的路径③来完成攻击。RFTM 攻击中客户端不会向服务器发出异常请求留下痕迹，使缓存中毒攻击更加隐蔽。

4.1 攻击流程

对于使用了应用程序缓存的站点，正常用户请求 manifest 文件常常是通过图 3 中标号为③的流程，RFTM 即是要达到通过该流程缓存恶意内容的效果。需要注意，该种应用程序缓存攻击需要具备 3.2.1 节所述攻击方法的前提条件，并且要求目标合法网站已经使用了应用程序缓存功能下面假设合法网站的域名为 www.domain.com，以图 1 中的 manifest 文件为合法文件（简称为文件 MA），攻击的主要步骤如下。

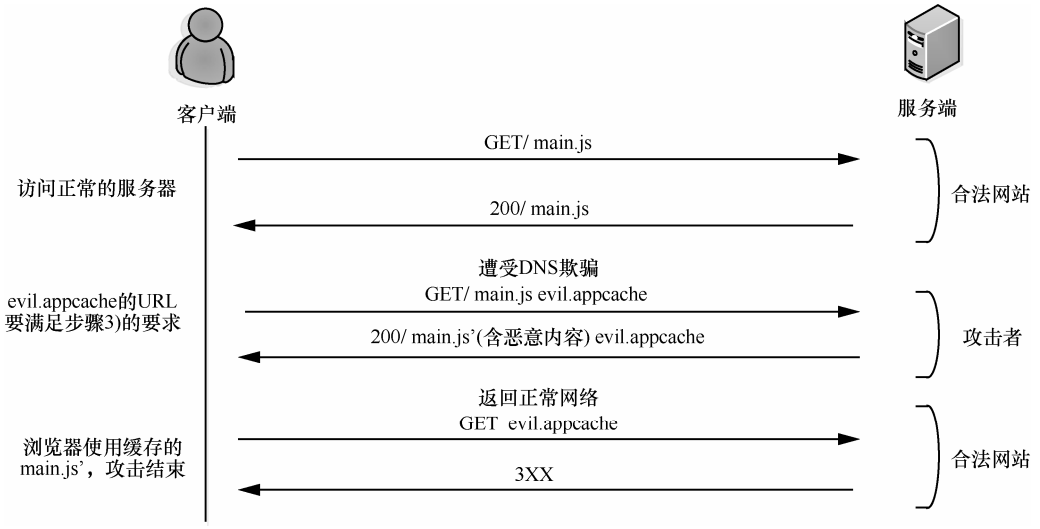

图 4 应用程序缓存中毒攻击示意

1) 攻击者从 www.domain.com 下载合法网站的 MA 文件。由于 manifest 文件是要交给浏览器在客户端处理的，所以很容易得到原 manifest 文件。

2) 复制 MA，并更改其内容确保会引发客户端更新，例如，将其中的 v1.0.0 更改为 v2.0.0。保持文件名不变，更改后的文件称为 MB，如图 5 所示。

```
1   CACHE MANIFEST
2   # 2015-09-15 v2.0.0
3   /logo.gif
4   /main.js
5
6   NETWORK:
7   login.asp
8
9   FALLBACK:
10  /html5/ /404.html
```

图 5　MB 文件

3) 攻击者模仿 www.domain.com 搭建恶意的 Web 服务器，包括相同外观的主页、URL 相同的 manifest 文件，以及需要缓存的 logo.gif、main.js 等文件，使受害者不易发现网页被替换。同时所有文件 URL 也要与原网站相同。

4) 在缓存文件中加入攻击代码，如窃取信息的 JavaScript，利用浏览器漏洞的 Shellcode 等。这些代码将可以长期保持在受害者的浏览器中。

5) 通过 DNS 欺骗、中间人等手段，在受害者访问的网页中加入透明的 iframe 标签，使受害者不知情地访问 www.domain.com，并收到攻击者创建的恶意网站内容。由于 MB 文件与原 MA 文件不同，所以浏览器会更新 www.domain.com 的缓存文件（不妨假设浏览器已经按照 MA 文件缓存）。

6) 接下来攻击者修改 MB 文件与 MA 文件相同，称为 MA'文件，本例中即将 v2.0.0 改回 v1.0.0。再通过步骤 5)中的方式，使受害者浏览器再次更新应用程序缓存，注意排除 HTTP 缓存对 manifest 文件的影响。这关键的一步使用户再次访问合法 www.domain.com 时不会更新应用程序缓存文件，直到 www.domain.com 本身对应用程序缓存文件进行了更新。

7) 攻击结束后，用户回到正常的网络环境中，再次访问 www.domain.com。由于 MA'与 MA 在内容、路径、文件名上完全一致，而浏览器发出的 GET 请求也与原来的正常请求一样，故不会触发更新缓存，从而使用带有攻击代码的缓存页，直到 www.domain.com 对 manifest 文件进行了更新，或者用户手动删除应用程序缓存。

RFTM 攻击如图 6 所示。

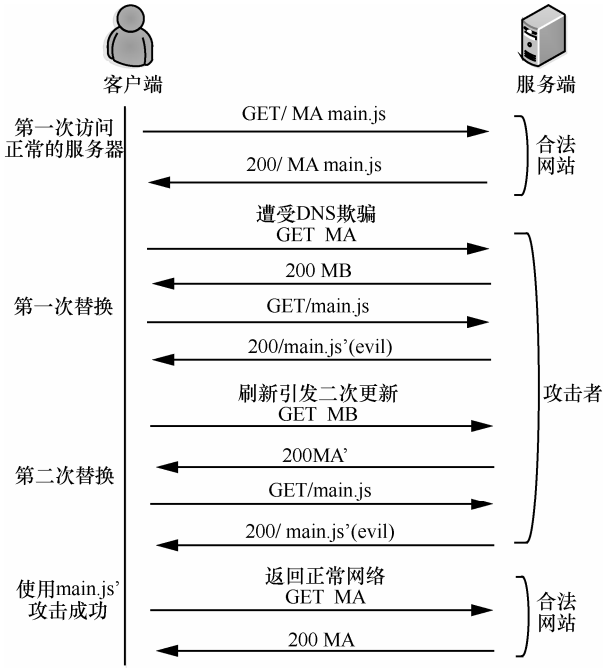

图 6　RFTM 攻击

4.2　攻击测试

本次攻击测试以小米公司的“一点资讯”[注1]站点为例。这是一个新闻资讯类站点，小米手机定制系统 MIUI 捆绑浏览器默认页面即有该站点的导航，以方便用户查看新闻资讯。该站点使用 HTML5 应用程序缓存，使手机用户在离线时也可以查看资讯摘要，减少了移动端流量，提供了良好的离线体验，其中的 cache.appcache 文件如图 7(a)所示。下面将针对该站点进行 RFTM 攻击测试。

攻击准备：攻击者首先制作与“一点资讯”功能外观相同的页面并加入攻击代码（这里用弹窗说明）；在主页相同目录准备好 manifest 文件 cache.appcache，内容如图 7(b)所示，作为第一次替换使用。另外，为了防止 HTTP 缓存对攻击的影响，在服务器的配置文件设置恰当的 expires 、last-modified、cache-control 头域。

攻击实施：攻击者与受害者同处一个 Wi-Fi 局域网。打开本机 Apache 服务器，使用中间人攻击综合工具 ettercap 对受害者进行 ARP 欺骗，并使用 etterfilter 在用户流量中加入不可见的内嵌窗口，让受害者不知情地访问“一点资讯”站点。

注1：http://news.browser.miui.com/。

```
1 CACHE MANIFEST
2 # M@lBBQWq.html
3 # _L78vQly.css
4 # 9HR3VSJs.js
5 //ssl-cdn.static.browser.mi-img.com/news/Do2ZfJr_.png
6
7 NETWORK:
8 *
```

(a) 原 manifest 文件

```
1 CACHE MANIFEST
2 # M@lBBQWq.html
3 # _L78vQly.css
4 # 9HR3VSJs.js
5 //ssl-cdn.static.browser.mi-img.com/news/Do2ZfJr_.png
6 # first replace
7 NETWORK:
8 *
```

(b) 替换文件

图 7　“一点资讯”manifest 文件

同时，对受害者进行 DNS 欺骗，使“一点资讯”的域名解析到攻击者的服务器。在受害者访问一次攻击者服务器后，攻击者即可修改 manifest 文件内容为图 7(a)中所示，与原 manifest 文件一致，待受害者再次被迫访问“一点资讯”时，2 次替换完成，RFTM 攻击结束。

攻击效果：在更换网络环境后，当受害者再次访问“一点资讯”站点时，浏览器弹出对话框，攻击成功。攻击者完全可以加入功能更加强大的恶意代码，在受害者每次使用“一点资讯”查看新闻时执行，直到站点更新 manifest 文件或用户手动删除应用程序缓存。

4.3　攻击对比分析

与 Kuppan[2]和 Homakov[16]提出的应用程序缓存中毒攻击相比，RFTM 攻击的主要特点如表 1 所示。就攻击条件而言， RFTM 和 Kuppan 的攻击都不要求攻陷 Web 站点，但需要对客户端进行流量劫持，而且 RFTM 还要求站点使用应用程序缓存，条件较为苛刻。就隐蔽性而言，在 Homakov 的攻击中，HTTP 缓存机制会使服务器原有流量减少，Kuppan 的攻击会请求不存在的 manifest 文件留下异常信息，而 RFTM 攻击使受害者在缓存恶意内容的同时，最终的 manifest 文件也与合法网站上的一致，当用户回到安全网络环境再次访问合法网站时，服务器端不会收到任何异常请求，通信流量与缓存中毒前完全相同，攻击更加隐蔽。因此，通过对服务器的配置无法防范 RFTM 攻击，该方法更加适合攻击者对大范围用户进行缓存投毒，而 Kuppan 的攻击则能够通过对服务器进行合理配置防范。就攻击成功后客户端恢复难易度而言，RFTM 和 Kuppan 的攻击均需用户手动清除离线数据，步骤较为复杂，而 Homakov 的攻击清除浏览器缓存即可恢复。

目前，Web 站点还没有普遍使用应用程序缓存，故 RFTM 攻击的影响范围不如 Kuppan 的攻击广泛。但随着逐渐增多的离线 Web 应用，如小米一点资讯、离线 Gmail、在线二元交易平台 Binary.com、Zoho 离线文档、Remember the Milk、WordPress 等，以及日益普及的移动端 HTML5 App，RFTM 应用程序缓存中毒攻击将会造成巨大的安全隐患。

表 1　各 AppCache 缓存中毒攻击对比

对比项	RFTM	Kuppan	Homakov
攻击条件	高	低	高
隐蔽性	高	低	低
服务端防范可行性	否	是	否
恢复难易	难	难	易

5　防御方案

本节讨论并总结针对应用程序缓存中毒攻击的防御方案。针对上述攻击，可以从 2 个方面出发来考虑防御措施。1) 阻止攻击发生的前提条件，如防止流量劫持；2) 从应用程序缓存的应用层面，如对服务器进行设置，引入防止替换缓存文件的签名机制。

5.1　通用防御策略

由于应用程序缓存中毒攻击需要首先劫持用户的 HTTP 通信，因此为了防止此类攻击，站点应该尽可能多地开启 HTTPS 来防止流量劫持，但是 sslstrip[22]的易于使用和用户自身较低的安全意识，使防护效果不佳。所以，更推荐使用 HTTP strict transport security（HSTS）[23]，只要用户第一次访问了合法的站点，浏览器在很长时间里都只能用 HTTPS 访问该站点，这可以较好地防御流量劫持，从而不受应用程序缓存攻击影响，然而目前只有极少数站点开启了 HSTS[9]。

需要注意，若攻击者具备了修改合法站点 Web

文件的权限，使用 HTTPS 并不能起到任何防御作用。另外，若不使用缓存，可以在 HTTP 头加入 no-store 来强制取消缓存，但同时这也会造成用户体验的负面影响。

5.2　应用层面防御方案

根据 Kuppan 的应用程序缓存中毒攻击，客户端会请求不存在的 manifest 文件，并期望获得使用本地应用程序缓存的 HTTP 响应码。针对这样的攻击特征，网站管理员可以设置对异常的 manifest 文件请求返回 404 状态码，从而取消客户端的应用程序缓存。同时，出于安全的考虑，浏览器在请求 manifest 文件收到错误的 MIME 类型及异常状态码时，也应取消应用程序缓存的使用。

而采用 RFTM 攻击方法，客户端则不会发出异常请求，服务端无从察觉，采用上述措施无法防御。故本文在应用层面提出一种轻量级签名防御方案 Sec-Cache，来防止攻击者替换 manifest 文件。

5.2.1　轻量级签名防御方案 Sec-Cache

攻击者替换 manifest 文件使用户缓存有害的内容，根源是浏览器没有对 manifest 文件验证来源。虽然可以采用目前常用的证书机制来对缓存机制签名，但由于会引入 CA 等，造成较大的安全开销。因此，本文提出一个通过服务器和客户端协作的轻量级签名方案，来防止攻击者对 manifest 文件的伪造。描述方案使用的符号定义如表 2 所示。

下面从服务器端和客户浏览器端这 2 方面来对签名方案加以说明。服务器端只在必要的时候执行初始化阶段，如生成密钥、更新 manifest 文件。服务器在每次收到客户端请求 manifest 文件时，执行签名阶段步骤。注意，服务器需要在密钥过期之前一段时间产生新的公私钥对，并在签名保护下发布新的公钥。

表 2　　签名方案使用的符号

符号	含义
pk	服务器当前的公钥
sk	服务器当前的私钥
m	符合 W3C 标准的 manifest 文件
Sig(·)	RSA 签名算法
Ver(·)	RSA 验证算法
H(·)	安全散列算法
timestamp	时间戳，GTM 1970-1-1 00:00:00 到当前时间的秒数
check	签名字段，以注释的形式写入 manifest 标准文件末尾
expire	公钥过期时间，格式同 *timestamp*

1) 服务端初始化阶段

①生成 RSA 算法的公私钥对(*pk*, *sk*)并妥善保存；

②编写符合 W3C 标准的 manifest 文件 *m*。

2) 服务器签名阶段

①计算当前时间戳 *timestamp*；

②计算 *check*=*Sig*(*sk*, *H*(*m*||*timestamp*||*pk*||*expire*))；

③收到客户端请求后发送 *m*||*pk*||*timestamp*|| *expire*||*check*。

3) 客户端校验阶段

①判断是否存储当前 URL 的 *pk*，若无，则存储服务器发布的(*pk*, *expire*)；

②若已存储密钥 *pk*，判断当前时间是否超过本地存储的过期时间 *expire*；

③根据 *timestamp* 判断消息是否为重放，若是，则拒绝该 manifest 文件；

④计算 *h*'=*H*(*m*||*timestamp*||*pk*||*expire*)；

⑤判断 *h*'是否等于 *Ver*(*pk*, *check*)，若等于，则接受该 manifest 文件。

签名方案执行流程如图 8 所示。

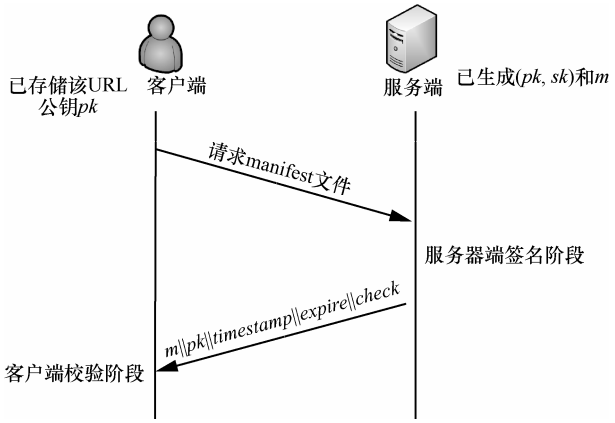

图 8　签名方案执行流程

为防止攻击者为不使用应用程序缓存的合法网站加入应用程序缓存，当客户端请求 manifest 文件收到 HTTP 404 状态码时，不应使用应用程序缓存；但在过期时间内客户端应不删除已有缓存及公钥，以防止攻击者删除原有缓存内容。

5.2.2　Sec-Cache 方案实现

为了验证方案的可行性并进行性能评估，实验在 apache 服务器上用 openssl 产生公私钥对，并采用 PHP 动态生成 manifest 文件的方法实现了服务器端签名；在客户端，以 IE 插件形式模拟了客户端的校验过程。

在 Sec-Cache 方案中，*check* 与 *timestamp* 是随

着服务器签名时间变化的内容，故将这 2 个字段以新增 HTTP 头域或者 cookie 的形式发送给客户端以保持兼容性，其余内容以注释按“#关键字：”开头的形式加入 manifest 文件中。Sec-Cache 方案定义的关键字如表 3 所示，要求开发者加入的注释内容不能与关键字冲突。#SEC_CACHEv1.0 表示方案的版本，方便未来改进升级；#DELETE 关键字为服务器提供了删除客户端应用程序缓存的支持，当客户端收到该关键字时删除当前应用程序缓存。

表 3 Sec-Cache 方案定义的关键字

关键字	含义
#SEC_CACHEv1.0	使用安全方案的版本标志
#PUBLIC_KEY	服务器当前的公钥
#TIMESTAMP	时间戳
#EXPIRE	密钥过期时间
#CHECK	签名
#NEW_PUBLIC_KEY	服务器新的公钥（可选）
#DELETE	取消应用程序缓存

5.2.3 方案评估

Sec-Cache 方案采用了签名机制，在攻击者无法获得私钥的情况下，对 manifest 文件进行任何更改，客户端都会在验证签名阶段发现，故可以有效地防止攻击者伪造合法网站的 manifest 文件，从而防御 RFTM 攻击。同时，签名机制能够阻止攻击者在任意 URL 伪造 manifest 文件，故也可以防范 Kuppan 的攻击。另外，即使攻击者攻陷了合法站点，若没有获得私钥，也不能对 manifest 文件进行伪造。

该方案还具有良好的兼容性。基于原有的应用程序缓存工作流程，没有增加客户端与服务端的通信步骤；新增不变关键字段以注释形式加入 manifest 文件，不影响现有的浏览器处理应用程序缓存。

在效率上，该方案增加了少量传输时的流量，以及服务端和客户端的校验计算过程。通过查看 HTTP 头部的 Content-Length 信息，可知添加的关键字固定增加通信流量约为 384 byte，主要由发布的公钥与 check 字段组成，不随原 manifest 文件大小的增加而增加。为了测试该方案时间效率影响的大小，实验采用 Intel i7 4712MQ 处理器、6 GB 内存的笔记本电脑作为客户端，2 GB 内存、双处理器的桥接模式 Linux 虚拟机作为服务器测试。通过使用浏览器开发人员工具的计时功能，可以观测出服务器签名的耗时；通过在客户端代码中加入计时器的方法，计算客户端校验阶段耗时，这 2 部分耗时之和即为方案的损失时间。通过增加 manifest 文件的长度，多次实验计算均值，得到如图 9 所示的损失时间。

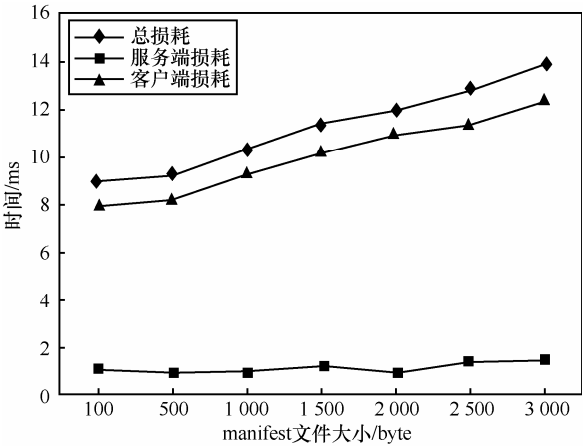

图 9 Sec-Cache 方案损失时间

由实验结果可知，服务器端签名阶段耗时基本稳定在 1~2 ms，表明方案对服务器的影响较小，损失时间主要在客户端。通常，一个 manifest 文件的大小不超过 1 000 byte，而完成一个大中型网页的加载少则需要几百毫秒的时间，相比于 Sec-Cache 方案所带来的优点，增加的开销微不足道。

由于服务器不能主动通知客户端，该方案在更新密钥时存在缺陷。若客户端在第一次保存了公钥后，在密钥过期时间到达之前没有及时更新密钥，则会删除原密钥脱离保护。即该方案只能保护第一次登录的是合法网站并且及时更新公钥的用户，这点与 HSTS 相似。在更高安全需求的场景，可以考虑结合 CA 证书机制。

6 结束语

HTML5 标准化后，越来越多的 Web 应用开始使用这项新标准。本文首先详细分析了 HTML5 应用程序缓存以及应用程序缓存中毒攻击，并讨论了攻击所产生的危害。针对使用应用程序缓存的站点，本文提出了更加隐蔽的 RFTM 攻击方式。通过 2 次替换 manifest 文件，攻击者缓存恶意内容的同时，最终的 manifest 文件也与合法网站上的一致。因此，合法服务器端不会收到任何异常的请求，使缓存中毒更加隐蔽，并且通过对服务器的配置无法防范。接下来，从传输层与应用层这 2 个方面，对

应用程序缓存中毒攻击的防御方案进行了讨论，并设计了一种轻量级签名防御方案 Sec-Cache 来防止 RFTM 应用程序缓存中毒攻击。实验表明该方案具有良好的效率与兼容性。

参考文献：

[1] LEENHEER N. How well does your browser support HTML5[EB/OL]. http://html5test.com.

[2] KUPPAN L. Attacking with HTML5[EB/OL]. http://www.andlabs.org.

[3] ZALEWSKI M. Geolocation spoofing and other UI woes[EB/OL]. http://seclists.org/bugtraq/2010/Aug/201.

[4] SON S, SHMATIKOV V. The postman always rings twice: attacking and defending postMessage in HTML5 Websites[C]//NDSS. 2013.

[5] 王晓强. 基于 HTML5 的 CSRF 攻击与防御技术研究[D]. 成都: 电子科技大学, 2013.
WANG X Q. Research of CSRF attack and defense techniques based on HTML5 [D]. Chengdu: University of Electronic Science and Technology of China, 2013.

[6] KULSHRESTHA A. An empirical study of HTML5 websockets and their cross browser behavior for mixed content and untrusted certificates[J]. International Journal of Computer Applications, 2013, 82(6): 13-18.

[7] JIN X, HU X, YING K, et al. Code injection attacks on HTML5-based mobile apps: characterization, detection and mitigation[C]// ACM SIGSAC Conference on Computer & Communications Security. 2014:66-77.

[8] GILGER J. Persistent AppCache injections [EB/OL]. https://heipei.github.io/2015/08/20/Persistent-AppCache-Injections/.

[9] JIA Y, CHEN Y, DONG X. Man-in-the-browser-cache: persisting https attacks via browser cache poisoning[J]. Computers & Security, 2015: 62-80.

[10] HANNA S, CHUL E, SHIN R, et al. The Emperor's new APIs: on the (in) secure usage of new client-side primitives[J]. W2sp Web Security & Privacy, 2010.

[11] 李潇宇, 张玉清, 刘奇旭,等. 一种基于 HTML5 的安全跨文档消息传递方案[J]. 中国科学院大学学报, 2013, 30(1):124-130.
LI X Y, ZHANG Y Q, LIU Q X, et al. Secure cross document messaging scheme based on HTML5 [J]. Journal of Graduate University of Chinese Academy of Sciences, 2013, 30(1):124-130.

[12] TIAN Y, LIU Y C, BHOSALE A, et al. All your screens are belong to us: attacks exploiting the HTMl5 screen sharing API[C]// Proceedings of the 2014 IEEE Symposium on Security and Privacy, IEEE Computer Society. 2014:34-48.

[13] HEIDERICH M, FROSCH T, JENSEN M, et al. Crouching tiger - hidden payload: security risks of scalable vectors graphics[C]// Proceedings of the 18th ACM Conference on Computer and Communications Security. 2011: 239-250.

[14] JOHNS M, LEKIES S, STOCK B. Eradicating DNS rebinding with the extended same-origin policy[C]//Usenix Conference on Security. 2013:621-636.

[15] LEE S, KIM H, KIM J. Identifying cross-origin resource status using application cache [C]//Proc NDSS '15. 2015.

[16] HOMAKOV E. Using AppCache and service worker for evil [EB/OL]. http://sakurity.com/blog/2015/08/13/middlekit.html.

[17] W3C. Offline Web applications–HTML5[EB/OL]. https://www.w3.org/TR/html5/browsers.html#offline.

[18] VALLENTIN M, BEN-DAVID Y. Persistent browser cache poisoning [R/OL].http://eecs.berkeley.edu/~yahel/papers/Browser-Cache-Poisoning.Song.Spring10.attack-project.pdf.

[19] WALIULLAH M, GAN D. Wireless LAN security threats & vulnerabilities: a literature review[J]. International Journal of Advanced Computer Science & Application, 2014, 5(1): 176-181.

[20] LAVA. Shell of the future: reverse web shell handler for XSS exploitation[EB/OL]. http://www.andlabs.org/tools/sotf/sotf.html

[21] LAVA. HTML5 based JavaScript network reconnaissance tool [EB/OL]. http://www.andlabs.org/tools/jsrecon.html.

[22] MARLINSPIKE M. A tool for exploiting moxie marlinspike's SSL "stripping"Attack[EB/OL]. https://github.com/ moxie0/sslstrip.

[23] Internet Engineering Task Force. HTTP strict transport security (HSTS) [S/OL]. https://tools.ietf.org/html/ rfc6797.

作者简介：

贾岩（1992-），男，河北石家庄人，西安电子科技大学博士生，主要研究方向为网络与系统安全。

王鹤（1987-），女，河南滑县人，博士，西安电子科技大学讲师，主要研究方向为信息系统安全与量子密码。

吕少卿（1987-），男，山西五寨人，西安电子科技大学博士生，主要研究方向为在线社交网络安全。

张玉清（1966-），男，陕西宝鸡人，博士，中国科学院大学教授、博士生导师，主要研究方向为网络与信息系统安全。