


# 详解Web缓存欺骗攻击

阅读量 18453 | 稿费 300

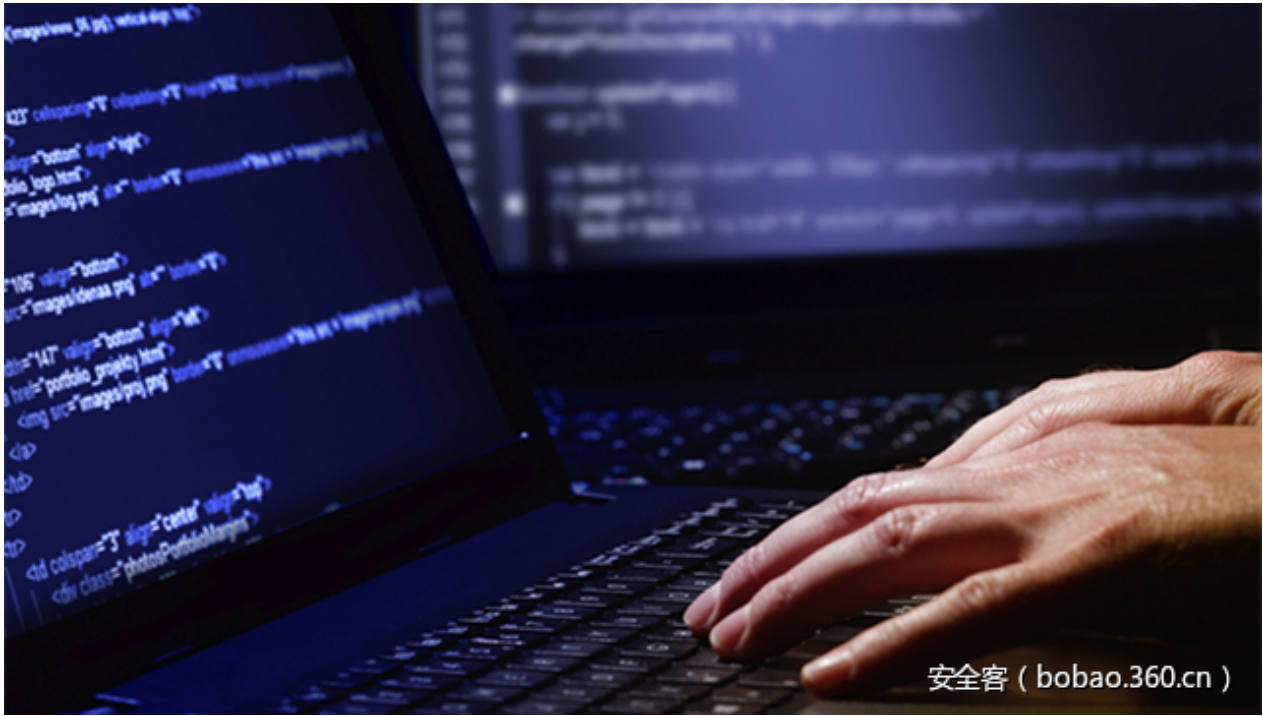
分享到：

发布时间：2017-07-31 14:03:19

## 译文声明

本文是翻译文章，文章原作者，文章来源：blackhat.com  
原文地址：<https://www.blackhat.com/docs/us-17/wednesday/us-17-Gil-Web-Cache-Deception-Attack-wp.pdf>

译文仅供参考，具体内容表达以及含义原文为准



译者：[興趣使然的小胃](#)

预估稿费：300RMB

投稿方式：发送邮件至linwei#360.cn，或登陆网页版在线投稿

## 一、摘要

Web缓存欺骗（Web Cache Deception）是一种新的web攻击方法，包括web框架以及缓存机制等在内的许多技术都会受到这种攻击的影响。攻击者可以使用这种方法提取web用户的私人及敏感信息，在某些场景中，攻击者利用这种方法甚至可以完全接管用户账户。

Web应用框架涉及许多技术，这些技术存在缺省配置或脆弱性配置，这也是Web缓存欺骗攻击能够奏效的原因所在。

如果某个用户访问看上去人畜无害、实际上存在漏洞的一个URL，那么该Web应用所使用的缓存机制就会将用户访问的具体页面以及用户的私人信息存储在缓存中。

## 二、背景介绍

### 2.1 什么是Web缓存

很多网站都会使用web缓存功能来减少web服务器的延迟，以便更快地响应用户的内容请求。为了避免重复处理用户的请求，web服务器引入了缓存机制，将经常被请求的文件缓存起来，减少响应延迟。



通常被缓存的文件都是静态文件或者公共文件，如样式表（css）、脚本（js）、文本文件（txt）、图片（png、bmp、gif）等等。通常情况下，这些文件不会包含任何敏感信息。许多指导性文章在提及web缓存的配置时，会建议缓存所有公开型静态文件，并忽略掉这些文件的HTTP缓存头信息。

有多种方法能够实现缓存，比如，浏览器端也可以使用缓存机制：缓存文件后，一段时间内浏览器不会再次向web服务器请求已缓存的文件。这类缓存与web缓存欺骗攻击无关。

实现缓存的另一种方法就是将一台服务器部署在客户端和web服务器之间，充当缓存服务器角色，这种实现方法会受到web缓存欺骗攻击影响。这类缓存有各种表现形式，包括：

- 1、**CDN（Content Delivery Network，内容分发网络）**。CDN是一种分布式代理网络，目的是快速响应内容请求。每个客户端都有一组代理服务器为其服务，缓存机制会选择离客户端最近的一个节点来提供服务。
- 2、**负载均衡（Load balancer）**。负载均衡除了能够通过多台服务器平衡网络流量，也能缓存内容，以减少服务器的延迟。
- 3、**反向代理（Reverse proxy）**。反向代理服务器会代替用户向web服务器请求资源，然后缓存某些数据。

了解了这些缓存机制后，让我们来看看web缓存的实际工作过程。举个例子，“http://www.example.com”配置了一个反向代理服务器作为web缓存。与其他网站类似，这个网站使用了公共文件，如图片、css文件以及脚本文件。这些文件都是静态文件，该网站的所有或绝大部分用户都会用到这些文件，对每个用户来说，此类文件返回的内容没有差别。这些文件没有包含任何用户信息，因此从任何角度来看，它们都不属于敏感文件。

某个静态文件第一次被请求时，该请求会直接穿透代理服务器。缓存机制没见过这个文件，因此会向服务器请求这个文件，然后服务器会返回文件内容。现在，缓存机制需要识别所接收的文件的类型。不同缓存机制的处理流程有所不同，但在大多数情况下，代理服务器会根据URL的尾部信息提取文件的扩展名，然后再根据具体的缓存规则，决定是否缓存这个文件。

如果文件被缓存，下一次任何客户端请求这个文件时，缓存机制不需要向服务器发起请求，会直接向客户端返回这个文件。

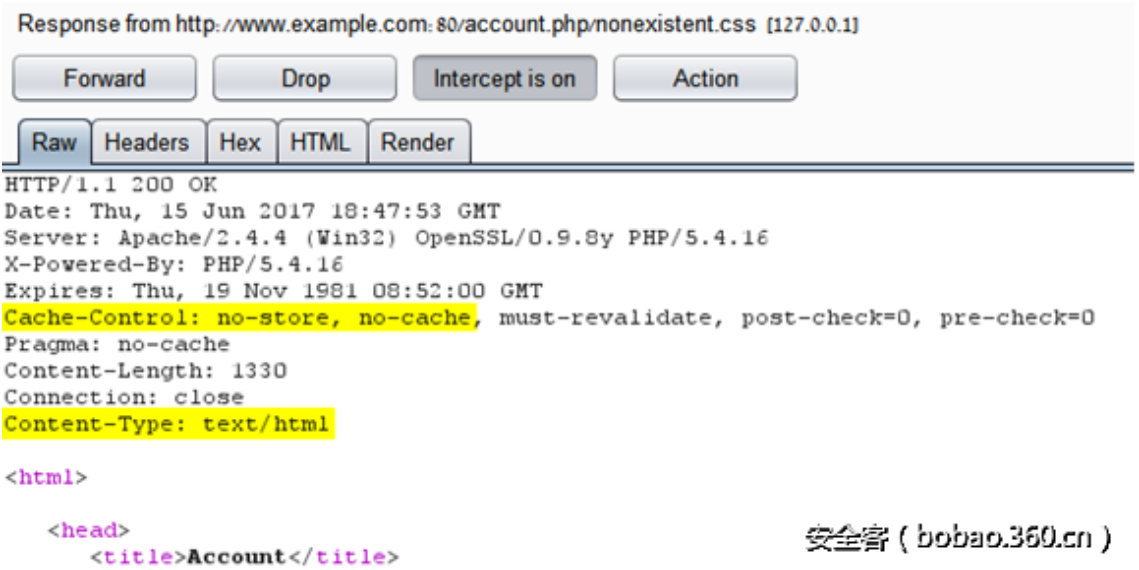
2.2 服务器的响应

Web缓存欺骗攻击依赖于浏览器以及web服务器的响应，这一点与RPO攻击类似，读者可以参考The Spanner[1]以及XSS Jigsaw[2]发表的两篇文章了解相关概念。

假设某个URL地址为“<http://www.example.com/home.php/nonexistent.css>”，其中home.php是一个真实页面，而非existent.css是个不存在的页面，那么当用户访问这个地址，会出现什么情况呢？

在这种情况下，浏览器会向该URL发送一个GET请求。我们比较感兴趣的是服务器的反应。取决于服务器的实现技术以及具体配置，web服务器可能会返回一个200 OK响应，同时返回home.php页面的内容，表明该URL与已有的页面一致。

服务器返回的HTTP响应头与home.php页面的响应头相同：即这两个响应头包含一样的缓存头部以及一样的内容类型（本例中内容类型为text/html），如下图所示：

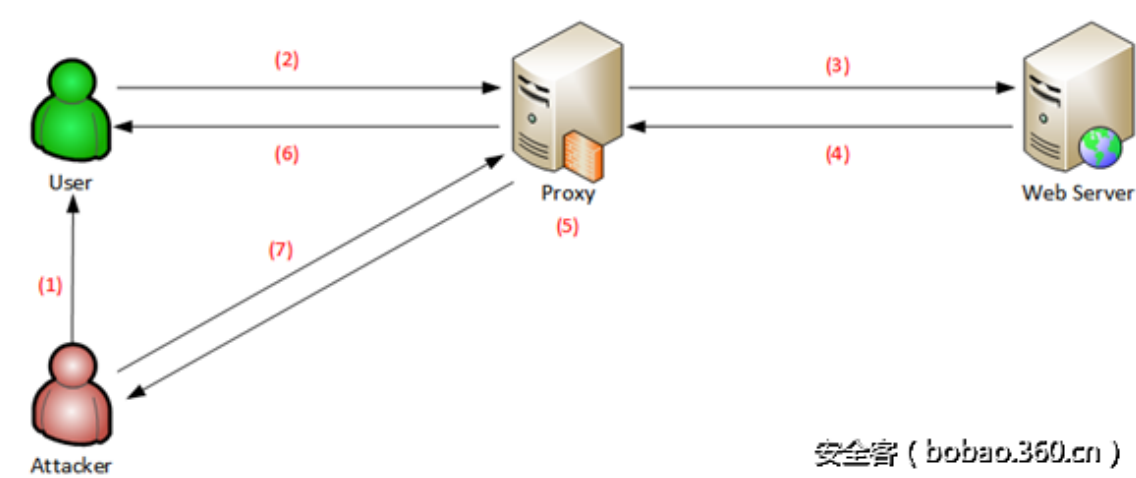


三、Web缓存欺骗方法

未经授权的攻击者很容易就能利用这个漏洞，具体步骤如下：

- 1、攻击者诱使用户访问“<https://www.bank.com/account.do/logo.png>”。

- 2、受害者的浏览器会请求“<https://www.bank.com/account.do/logo.png>”。
- 3、请求到达代理服务器，代理服务器没有缓存过这个文件，因此会向web服务器发起请求。
- 4、Web服务器返回受害者的账户页面，响应代码为200 OK，表明该URL与已有页面一致。
- 5、代理机制收到文件内容，识别出该URL的结尾为静态文件扩展名（.png）。由于在代理服务器上已经设置了对所有静态文件进行缓存，并会忽略掉缓存头部，因此伪造的.png文件就会被缓存下来。与此同时，缓存目录中会创建名为“account.do”的一个新的目录，logo.png文件会缓存在这个目录中。
- 6、用户收到对应的账户页面。
- 7、攻击者访问“<https://www.bank.com/account.do/logo.png>”页面。请求到达代理服务器，代理服务器会将已缓存的受害者账户页面发给攻击者的浏览器。



四、攻击意义

如果攻击成功，那么包含用户私人信息的存在漏洞的页面就会被缓存下来，可以被公开访问。被缓存的文件是一个静态文件，攻击者无法冒充受害者的身份。该无文件无法被覆盖，直到过期之前仍然有效。

如果服务器的响应内容中包含用户的会话标识符（某些场景中会出现这种情况）、安全应答、CSRF令牌等信息，那么攻击造成的后果将会更加严重。这种情况下，攻击者可以借助其他攻击手段最终完全掌控受害者账户。

五、攻击条件

攻击者若想实施Web缓存欺骗攻击，必须满足如下3个条件：

- 1、当访问如“<http://www.example.com/home.php/nonexistent.css>”之类的页面时，服务器需要返回对应的home.php的内容。
- 2、Web应用启用了Web缓存功能，并且会根据文件的扩展名来缓存，同时会忽略掉任何缓存头部。
- 3、受害者在访问恶意URL地址时必须已经过认证。

六、现有的Web框架

此类攻击是否能奏效，其中一个因素涉及到Web应用对特定URL的处理过程，这类URL由一个合法的URL以及尾部一个不存在的文件构成，如“<http://www.example.com/home.php/nonexistent.css>”。

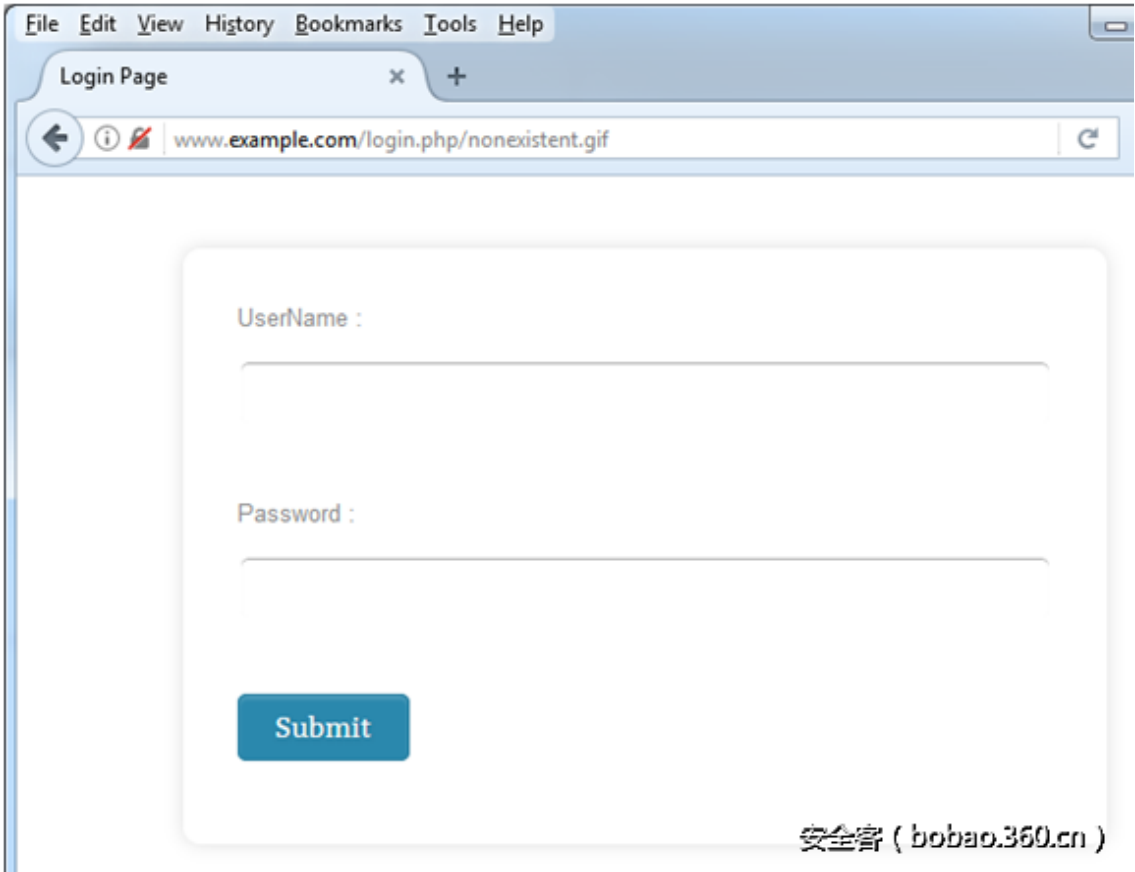
在这一部分内容中，我们会以具体的例子，向大家演示如何针对现有的几种web框架实施web缓存攻击，同时也会解释这些框架的具体配置及工作流程。

6.1 PHP

如果我们创建一个“纯净版”的PHP Web应用，没有使用任何框架，那么该应用会忽略掉URL尾部的任何附加载荷，返回真实页面的内容，并且响应代码为200 OK。



比如，当用户访问“http://www.example.com/login.php/nonexistent.gif”时，Web应用会返回login.php的内容，这意味着此时发起攻击的第1个条件已经得到满足。



6.2 Django

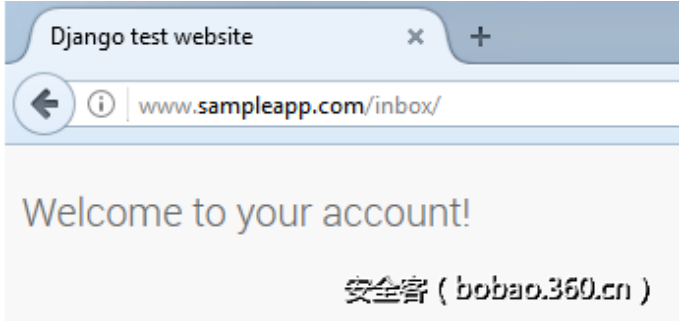
Django使用调度器（dispatcher）来处理Web请求，调度器使用urls文件来实现。在这些文件中，我们可以设置正则表达式来识别URI中具体请求的资源，然后返回对应的内容。

```
from django.conf.urls import include,url
from . import views

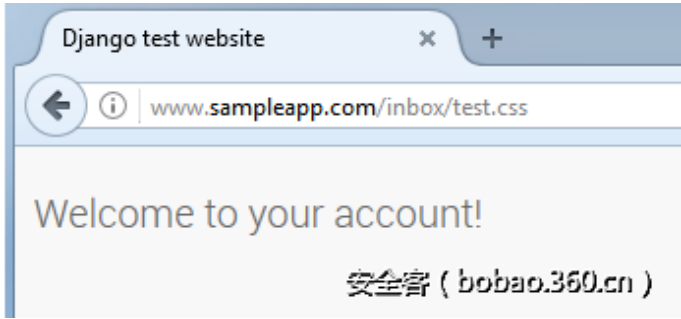
urlpatterns = [
    url(r'^inbox/', views.index, name='index')
]
```

安全客 (bobao.360.cn)

上图是Django的常见配置，根据这个配置，当客户端请求“http://www.sampleapp.com/inbox/”时，服务器会返回Inbox页面的内容。



如果将某个不存在的文件附加到该URL尾部（如“http://www.sampleapp.com/inbox/test.css”），这种正则表达式同样会匹配成功。因此，Django同样满足发起攻击的第1个条件。



此外，如果正则表达式忽略掉“inbox”尾部的斜杠，那么这种表达式也存在漏洞。

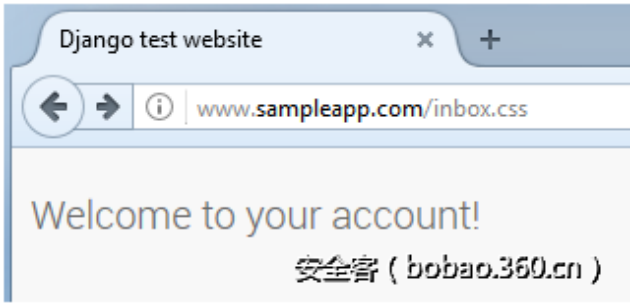
```
from django.conf.urls import include,url
from . import views

urlpatterns = [
    url(r'^inbox', views.index, name='index')
]
```

安全客 (bobao.360.cn)



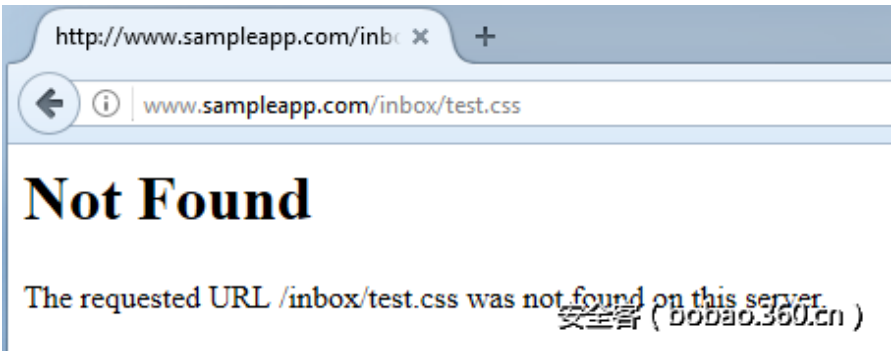
这种正则表达式不仅会匹配正常的URL（即“<http://www.sampleapp.com/inbox>”），也会匹配不存在的URL（如“<http://www.sampleapp.com/inbox.css>”）。



如果正则表达式尾部使用了“\$”符，那么就不会匹配这种恶意URL地址。

```
from django.conf.urls import include,url
from . import views

urlpatterns = [
    url(r'^inbox/$', views.index, name='index')
]
```



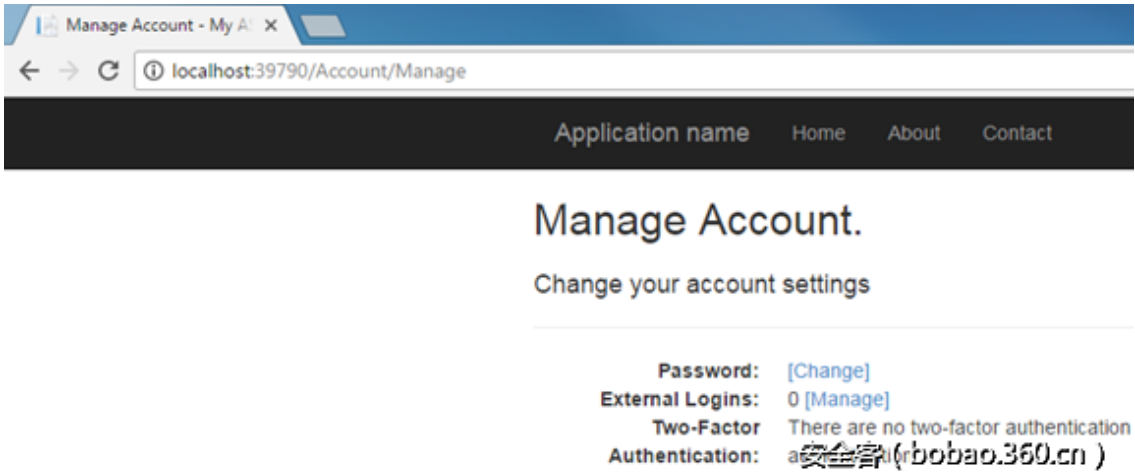
6.3 ASP.NET

ASP.NET框架中有个内置的功能，叫做FriendlyURLs，这个功能的主要目的是使URL看起来更加“整洁”同时也更加友好。当用户访问“<https://www.example.com/home.aspx>”时，服务器会删掉尾部的扩展名，将用户重定向至“<https://www.example.com/home>”。

我们可以在Route.config文件中配置这个功能，在ASP.NET应用中，这个功能默认情况下处于启用状态。

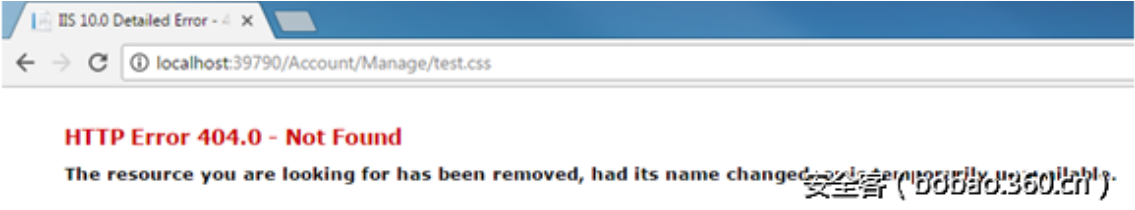
```
7 namespace WebApplication7
8 {
9     public static class RouteConfig
10    {
11        public static void RegisterRoutes(RouteCollection routes)
12        {
13            var settings = new FriendlyUrlSettings();
14            settings.AutoRedirectMode = RedirectMode.Permanent;
15            routes.EnableFriendlyUrls(settings);
16        }
17    }
18 }
```

启用FriendlyURLs功能时，当用户通过“<http://localhost:39790/Account/Manage.aspx>”地址访问已有的Manage.aspx页面时，服务器会移除.aspx扩展名，显示页面内容。



在这种配置下，当用户访问“<http://localhost:39790/Account/Manage.aspx/test.css>”时，.aspx扩展名会被移除，用户会被重定向到“<http://localhost:39790/Account/Manage/test.css>”，此时服务器会返回404错误。这意味着当ASP.NET启用FriendlyURLs功能时，攻击条件无法满足。



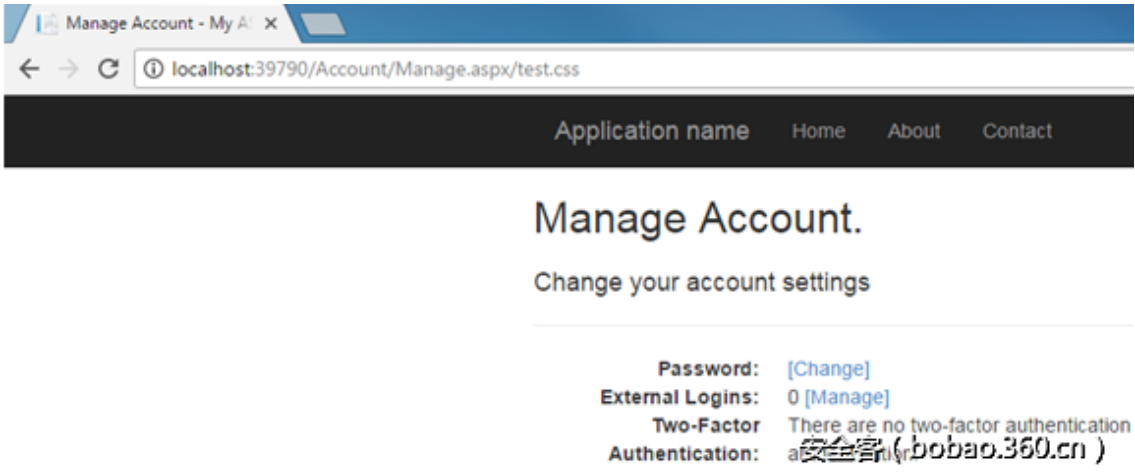


虽然FriendlyURLs默认处于启用状态，但很多网站并没有使用这个功能。该功能可以在Route.config文件中关闭。

```
7 namespace WebApplication7
8 {
9     public static class RouteConfig
10    {
11        public static void RegisterRoutes(RouteCollection routes)
12        {
13            var settings = new FriendlyUrlSettings();
14            settings.AutoRedirectMode = RedirectMode.Off;
15            routes.EnableFriendlyUrls(settings);
16        }
17    }
18 }
```

安全客 ( bobao.360.cn )

关闭该功能后，访问攻击URL地址时服务器会返回200 OK响应，并且会返回Manage.aspx页面的内容。



## 七、现有的缓存机制

攻击的第2个条件是web应用启用了Web缓存功能，并且会根据文件的扩展名来缓存，同时会忽略掉任何缓存头部。下面我们会以现有的某些缓存机制为例，介绍这些机制的缓存过程以及它们如何识别接收到的文件的类型。

### 7.1 Cloudflare

当来自web服务器的文件到达Cloudflare时，文件会经过两个阶段的处理过程。第一个阶段名为资格阶段（Eligibility Phase），此时Cloudflare会检查目标站点是否设置了缓存功能，也会检查文件来源目录是否设置了缓存功能。如果检查通过（检查基本都会通过，这也是为什么网站一开始就使用Cloudflare服务的原因所在），那么Cloudflare服务器就会检查具体的URL地址是否以如下静态扩展名结尾：

class, css, jar, js, jpg, jpeg, gif, ico, png, bmp, pict, csv, doc, docx, xls, xlsx, ps, pdf, pls, ppt, pptx, tif, tiff, ttf, otf, webp, woff, woff2, svg, svgz, eot, eps, ejs, swf, torrent, midi, mid

如果URL地址的确以上述扩展名结尾，那么文件就会到达第二阶段的处理过程，即失格阶段（Disqualification Phase），此时Cloudflare服务器会检查HTTP缓存头部是否存在。

不幸的是，当我们访问恶意URL地址时，web服务器会返回已有的动态页面的缓存头部，这意味着服务器很有可能会返回带有“no-cache”指令的文件。

幸运的是，Cloudflare存在一个名为“边缘缓存过期TTL（Edge cache expire TTL）”的功能，这个功能可以用来覆盖任何已有的头部信息。将该功能设置为启用（on）状态时，服务器返回的带有“no-cache”指令的文件仍会被缓存下来。出于各种原因，在Cloudflare的建议下，该功能通常会处于启用状态。



Create a Page Rule for webcachedeception.com

If the URL matches: By using the asterisk (\*) character, you can create dynamic patterns that can match many URLs, rather than just one. [Learn more here](#)

Then the settings are:

Cache Level

Standard

Edge Cache TTL

2 hours

安全客 ( bobao.360.cn )

7.2 IIS ARR

应用程序请求路由（Application Request Routing，ARR）模块可以为IIS带来负载均衡功能。

ARR模块提供的一个功能就是缓存功能。Web服务器可以通过负载均衡器设置缓存规则，以便将文件保存到缓存目录中。在创建新的缓存规则时，我们使用通配符和目标扩展名来定义待缓存的文件类型。当文件经过ARR处理时，ARR会根据文件对应的URL来匹配缓存规则。实际上，ARR会根据URL尾部的扩展名来识别文件类型。

此外，IIS ARR中还包含一个选项，可以忽略掉文件的缓存头部，导致该规则在任何情况下都适用。

Edit Cache Control Rule

Apply rule:

Always

☐ Do not cache

☒ Cache

Cache duration (minutes):

60

Host name:

Example: www.contoso.com

URL:

\*.css

Example: \*/images/\*.jpg

OK

Cancel

安全客 ( bobao.360.cn )

如下图这个例子中，IIS ARR与两个web服务器相连接，并且根据配置会缓存所有的样式表和JavaScript文件。

Cache Control Rules

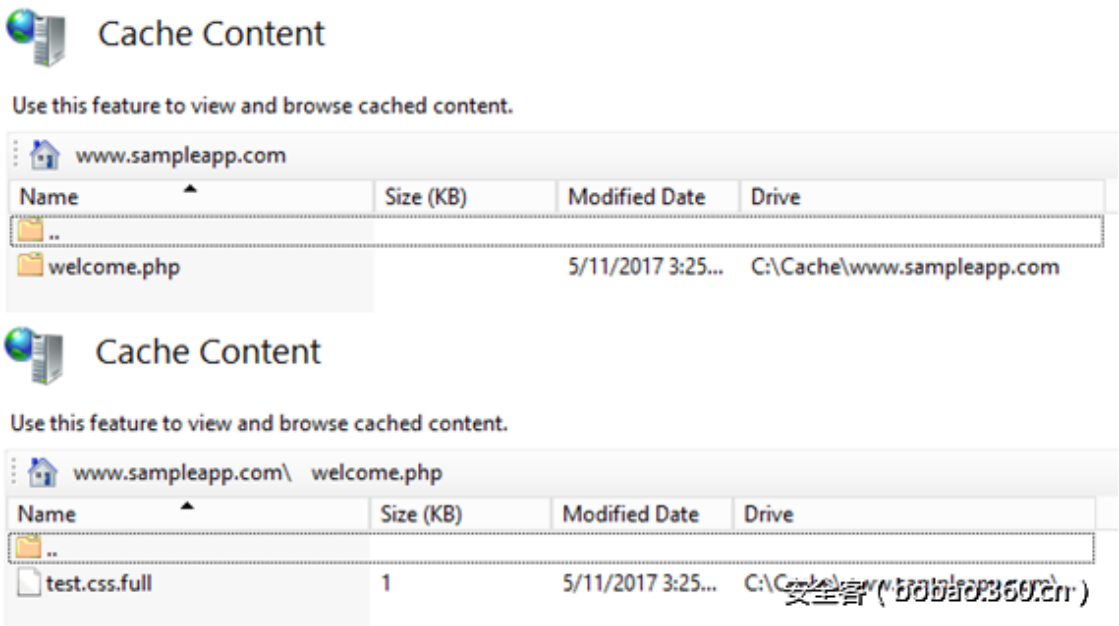
Use this feature to configure rules to manage cache control directives.

Cache	Host Name	URL	Condition	Cache Duration
Yes		*.css	Always	60
Yes		*.js	Always	10000

安全客 ( bobao.360.cn )

如果客户端访问恶意URL（<http://www.sampleapp.com/welcome.php/test.css>），那么缓存目录中就会生成一个新的目录，目录名为welcome.php，在该目录中，会生成名为test.css的一个新的文件，该文件的内容为用户访问的welcome.php页面的内容。





7.3 NGINX

作为负载均衡服务器，NGINX服务器也可以提供缓存功能，来缓存从web服务器返回的页面。

我们可以通过NGINX配置文件来配置缓存规则。如果使用下图所示的配置文件，那么NGINX就会缓存特定类型的静态文件，并且会忽略这些文件的缓存头部。

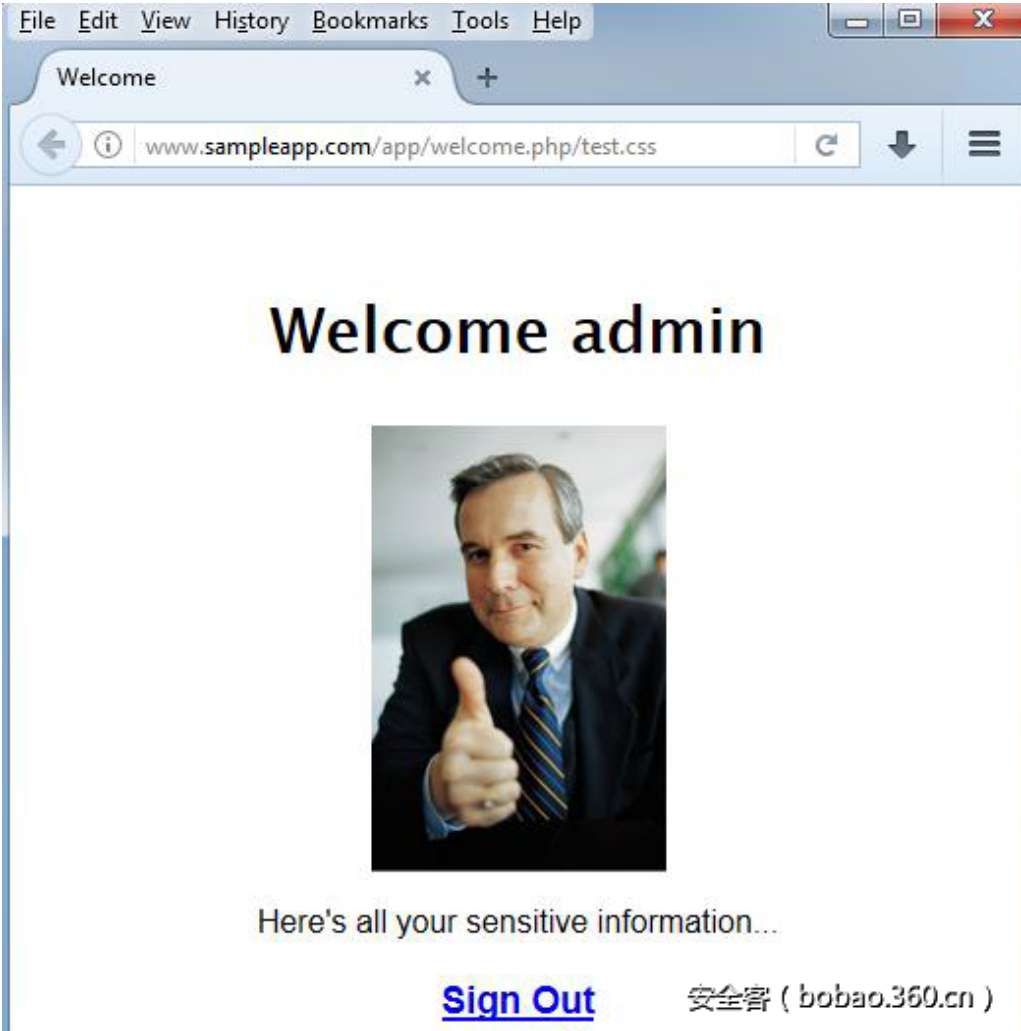
```
location ~* \.(css|js|gif|png)$ {
    proxy_cache my_cache;
    proxy_cache_valid 200 60m;
    proxy_pass http://[redacted];
    proxy_ignore_headers Expires Cache-Control Set-Cookie;
}
```

当来自于web服务器的某个页面到达NGINX时，NGINX会搜索URL尾部的扩展名，根据扩展名识别文件的类型。

首先，缓存目录中没有缓存任何文件。

```
root@[redacted]:/cache# tree
└── temp
1 directory, 0 files
```

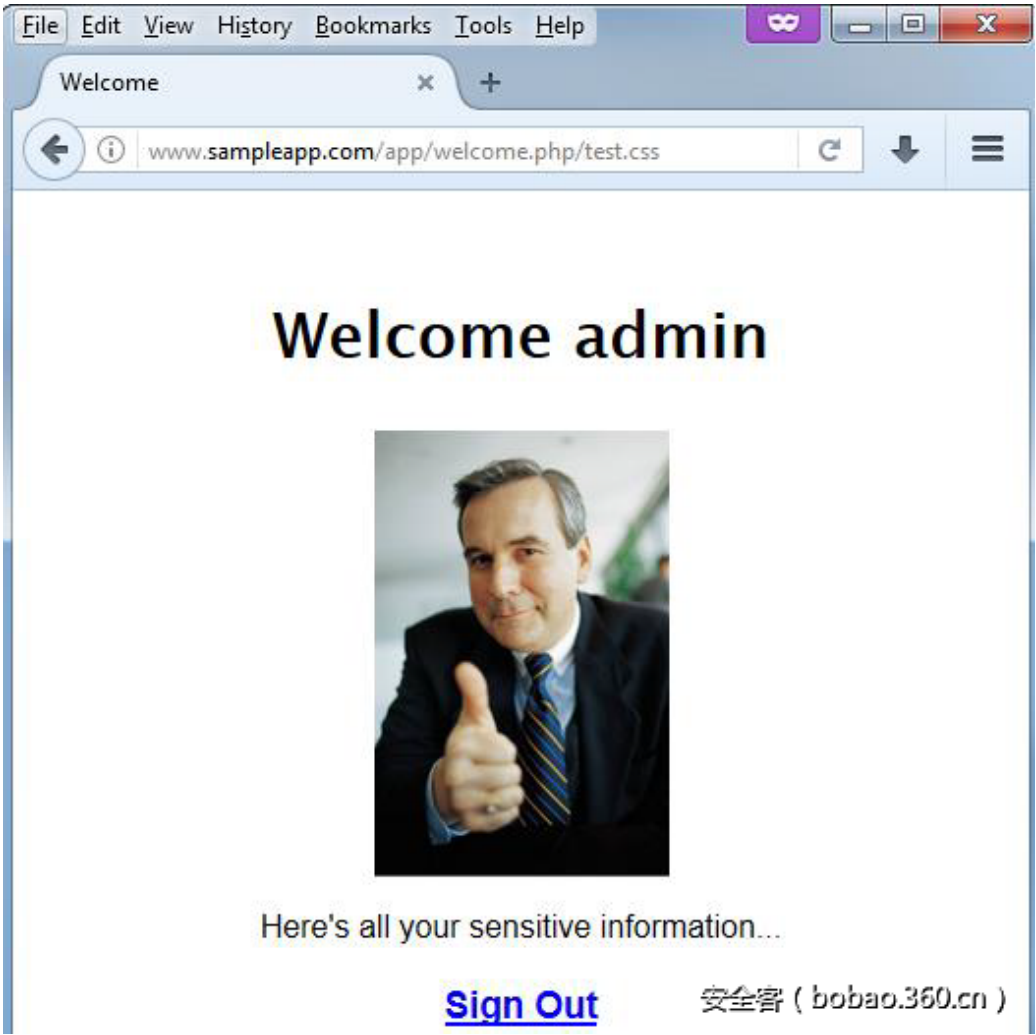
当经过认证的用户访问恶意URL时（http://www.sampleapp.com/app/welcome.php/test.css），用户的页面就会被缓存到缓存目录中。







接下来，未经认证的攻击者会访问恶意URL，此时NGINX服务器会返回已缓存的文件，文件中包含用户的隐私数据。



### 八、缓解措施

可以使用以下几种方法缓解此类攻击。

- 1、配置缓存策略，只有当文件的HTTP缓存头部允许缓存时，才会缓存这些文件。
- 2、将所有的静态文件保存到某个指定目录，并且只缓存这个目录。
- 3、如果缓存组件允许的话，需要将其配置为根据文件的具体内容来缓存文件。
- 4、配置web服务器，使其在处理诸如“http://www.example.com/home.php/nonexistent.css”的页面时，不会返回home.php的内容，而会返回404或者302响应。

### 九、总结

Web缓存欺骗攻击实施起来没有那么容易，但依然可以造成严重的后果，包括泄露用户的隐私信息、攻击者可以完全控制用户的账户等等。此前我们发现一些知名的网站会受到此类攻击影响，并且这些网站中绝大部分由最为常见的CDN服务商来提供服务。我们有理由相信，此时此刻仍有许多网站会沦为这类攻击的受害者。

虽然这份白皮书中只提到了可以满足web缓存欺骗攻击条件的几种技术，但还有其他许多web框架以及缓存机制存在脆弱性，攻击者可以使用类似技术发起攻击。

如果Web框架以及缓存机制可以创造条件来满足漏洞场景，那么我们认为这些Web框架及缓存机制本身并没有存在这类漏洞，它们的主要问题是脆弱性配置问题。

为了防御web缓存欺骗攻击，技术人员首先应当了解此类攻击发起的条件。此外，厂商应该有所作为，避免他们的产品符合攻击条件。以上要求可以通过禁用特定功能、更改默认设置及行为、提供警报信息以增强技术人员的警觉意识来实现。



## 十、致谢

感谢Sagi Cohen、Bill Ben Haim、Sophie Lewin、Or Kliger、Gil Biton、Yakir Mordehay、Hagar Livne。

## 十一、参考资料

[1] RPO – The Spanner 博客。

<http://www.thespanner.co.uk/2014/03/21/rpo/>

[2] RPO gadgets – XSS Jigsaw 博客

<http://blog.innerht.ml/rpo-gadgets/>

[3] Django URL分发器

<https://docs.djangoproject.com/en/1.11/topics/http/urls/>

[4] NGINX缓存机制

<https://serversforhackers.com/c/nginx-caching>

[5] Web缓存欺骗攻击

<http://omergil.blogspot.co.il/2017/02/web-cache-deception-attack.html>

[6] 针对PayPal主页的web缓存欺骗攻击

<https://www.youtube.com/watch?v=pLte7SomUB8>

[7] Cloudflare blog的参考资料

<https://blog.cloudflare.com/understanding-our-cache-and-the-web-cachedeception-attack/>

[8] Akamai博客上的参考资料

<https://blogs.akamai.com/2017/03/on-web-cache-deception-attacks.html>

本文翻译自 blackhat.com，[原文链接](#)。如若转载请注明出处。

安全知识



赞



收藏



興趣使然的小胃

分享到：



### 推荐阅读

