时间戳防盗链

最近更新时间：2020-07-17 10:38:48

1. 简介

CDN 支持多种访问控制；其中，时间戳防盗链可以通过对时间有关的字符串进行签名，将时间、签名信息通过一定的方式传递给 CDN 边缘节点服务器进行鉴权，从而正确响应合法请求、拒绝非法请求。相比于 referer 防盗链，时间戳防盗链的安全性更强。

1.1 原理说明

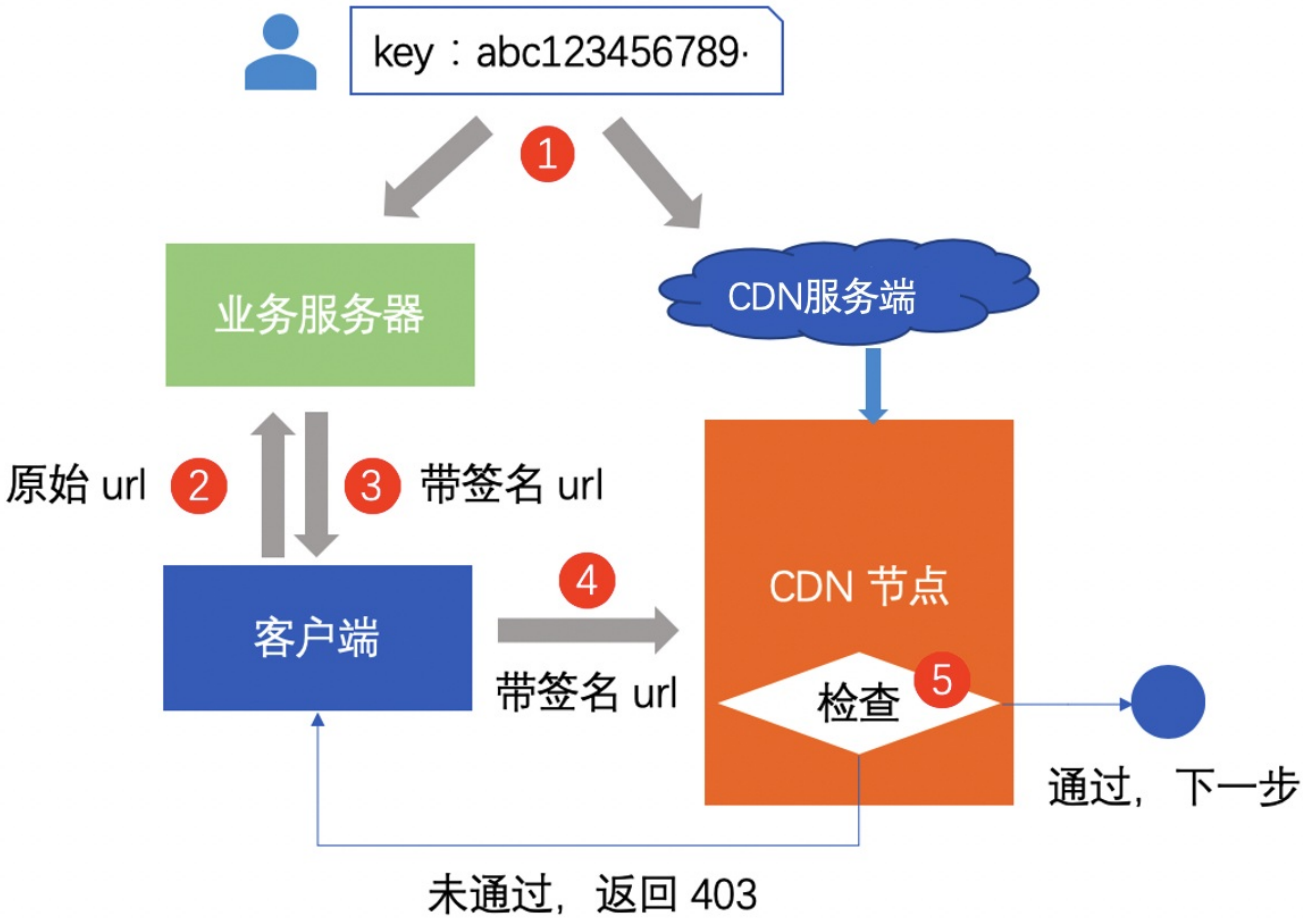
时间戳防盗链的目的是使得每个请求的 url 都具有一定的 **“时效性”**，所以 url 本身需要携带过期时间相关的信息，同时还需要确保这个过期时间不能被恶意修改，因此采用 md5 算法，将 **key、过期时间、文件路径**等信息进行加密得到签名加入 url，并在 CDN 节点进行验证。

1.2 鉴权过程

整个时间戳防盗链的实现需要以下几个部分配合：

- 客户端：负责发送原始请求给客业务服务器以及发送带时间戳加密的 url 给 CDN 节点进行验证；
- 源站业务服务器：根据约定的算法生成带签名参数的 url 返回给客户端；
- CDN 节点：负责对客户端进行时间、签名校验。

主要分为以下几个步骤：



（1）用户管理员CDN 控制台配置 key，并将 key配置进业务服务器。（2）当客户端请求资源时，将原始 url 发送至业务服务器。（3）业务服务器根据计算逻辑，将带有时间戳签名的 url 返回至客户端。（4）客户端使用带有时间戳签名的 url 请求资源。（5）CDN 检查 url 签名的合法性。

注意：若同时配置了 referer 防盗链、IP 黑白名单、时间戳防盗链，有一项不满足条件，即为不通过，响应 403。

2.配置步骤

2.1 开始配置

（1）登录 CDN 控制台，选择【域名管理】。（2）进入域名配置界面，**访问控制**模块下的**时间戳防盗链**点击【修改配置】进行时间戳防盗链的配置。

2.2 配置 KEY

key 即用于加密 url 的密文，需要填写主要和备用 key，两个 key 不能相等，一般来说使用其中一个即可。用户可以点击【随机生成】生成 key；也可以自行使用算法生成，如下方生成实例中 gen_key（）函数。

获取 key 以后，需要将 key 配置到业务服务器中，并实现相关签算逻辑。具体实现方式可参考算法说明及代码生成实例。

2.3 校验时间戳 URL

为保证服务器正确实现了签算逻辑，防止因签算逻辑错误导致 403 错误，在正式开启前必须进行 url 的正确性校验。

在业务服务器实现相关签算逻辑后，可获取一个生成的带有时间戳签名的 url 粘贴至下面输入框。

若提示错误，请仔细阅读算法说明进行错误检查。

> 时间戳 URL 计算器：为方便用户辅助检查，CDN平台提供一个 URL 时间戳签名计算器，请确保您的业务服务器可以与该时间戳计算器获得一致的结果。**但需注意：此计算器仅用于辅助检查，请勿为通过检验直接将计算粘贴至检验框中。**

2.4 确认开启时间戳防盗链

点击【确定】即可正式开启时间戳防盗链功能。 **在正式开启前，请务必确认业务服务器已经可以正确签算时间戳签名 url ，保证所有访问 url 都正确携带相关参数，**

3.算法说明

3.1 签名算法

- key：在开启时间戳防盗链时，可以在控制台上输入或随机生成。
- path：访问资源的 url 中的路径部分，例如：访问的 url 为 `http://xxx.yyy.com/DIR1/dir2/vodfile.mp4?v=1.1`，则 `path = /DIR1/dir2/vodfile.mp4`（注意不含 querystring 部分）。
- T：url 过期时间,当超过设置的过期时间时，该鉴权失效。按 unix_time 的 16 进制小写形式表示。 如 `2015-08-01 00:00:00 -> 1438358400 -> 55bb9b80` 。**注意，请务必使用 16 进制时间，若没有转化为 16 进制形式直接使用 unix_time, 将会被 CDN 认为是一个很大的过期时间，导致无法起到时间戳鉴权的作用。**

签名原始字符串 `S = key + url_encode(path) + T` 。 签名 `SIGN = md5(S).to_lower()`，`to_lower` 指将字符串转换为小写；

** 注：本文提到的 url_encode 算法，都是斜线不参与编码的 utf-8 编码，下同。**

3.2 签名参数传递方式

`<SIGN>` 和 `<T>` 作为URL查询参数进行传递。 - 签名参数 sign、 t，sign 在前，t 在后； - `<SIGN>`、 `<T>` 替换为对应的值， 实际url中不含<>；

例如: 原始访问的 url 为： `http://xxx.yyy.com/DIR1/dir2/vodfile.mp4?v=1.1` 最终形成的访问 url 为： `http://xxx.yyy.com/DIR1/dir2/vodfile.mp4?v=1.1&sign=<SIGN>&t=<T>`

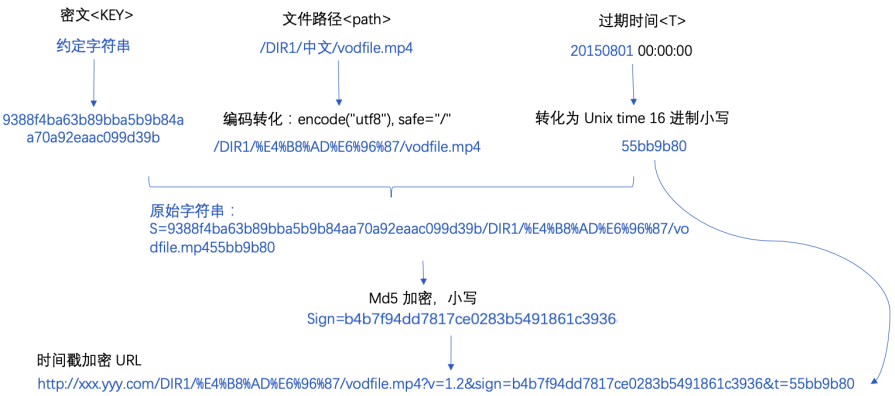
3.3 访问 url

访问 url 的 path 部分也需要 url_encode，其算法与签名时使用 url_encode 算法一致。斜线 / 不编码。

访问 url 为：

```
scheme + "://" + host + url_encode(path) + query_part
```

访问 url 生成示例



下图给出了一个访问 url 的生成示例。 原始 url : http://xxx.yyy.com/DIR1/中文/vodfile.mp4?v=1.2。

资源过期时间为：2015年8月1日。

参数	值
path	/DIR1/中文/vodfile.mp4
KEY	9388f4ba63b89bba5b9b84aa70a92eaac099d39b
T	55bb9b80

签名原始字符串 `S = key + url_encode(path) + T`

`S = 9388f4ba63b89bba5b9b84aa70a92eaac099d39b/DIR1/%E4%B8%AD%E6%96%87/vodfile.mp455bb9b80`

`SIGN = b4b7f94dd7817ce0283b5491861c3936`

`T=55bb9b80`

访问 url 为：

`http://xxx.yyy.com/DIR1/%E4%B8%AD%E6%96%87/vodfile.mp4?v=1.2&sign=b4b7f94dd7817ce0283b5491861c3936&t=55bb9b80`

4. 生成实例

python版

```
 #! /usr/bin/env python
#coding:utf-8

from hashlib import md5
import urllib
import sys
import time
from urlparse import urlparse, parse_qs
import traceback
import uuid
import base64

# 要求正确的 url_encode 编码，斜线 / 不编码；
# 井号 # 等在浏览器会直接识别为其它含义，若在 path 中必须编码；
# 问号 ? 等在 url 中有特殊含义，若在 path 中必须编码；
# 部分字符 "~!$&'()*+,:;=@[]" 不含双引号 "，虽有特殊含义，但在 path 部分，编码与否，都可以正常访问；
# 另一些，如 双引号 "，空格 " "，汉字等必须编码；
#
# 建议 url 的 path 中尽量不含上述部分，建议 url 中尽量不含上述部分。
"
```

```
#
# 参考
# https://www.wikiwand.com/zh-cn/%E7%99%BE%E5%88%86%E5%8F%B7%E7%BC%96%E7%A0%81
# https://www.wikiwand.com/en/Percent-encoding
# https://www.wikiwand.com/de/URL-Encoding
def url_encode(s):
    return urllib.quote(s.decode(sys.stdin.encoding).encode("utf8"), safe="/")

def to_deadline(rang):
    return int(time.time()) + rang

def t16(t):
    return hex(t)[2:].lower()   # 16 进制小写形式

def summd5(str):
    m = md5()
    m.update(str)
    return m.hexdigest()

def sign(key, t, path):
    a = key + url_encode(path) + t
    print("S: " + a)
    sign_s = summd5(a).lower()
    sign_part = "sign=" + sign_s + "&t=" + t
    return sign_part

def sign_url(key, t, p_url):
    url = urllib.unquote(p_url)
    up = urlparse(url)
    path = up.path
    sign_part = sign(key, t, path)
    p_query = up.query
    if p_query:
        query_part = "?" + p_query + "&"+ sign_part
    else:
        query_part = "?" + sign_part

    return up.scheme + "://" + up.netloc + url_encode(path) + query_part

def printurl_encode_help():
    print '''
# 要求正确的 url_encode 编码，斜线 / 不编码;
# 井号 # 等在浏览器会直接识别为其它含义，若在 path 中必须编码;
# 问号 ? 等在 url 中有特殊含义，若在 path 中必须编码;
# 部分字符 "~!$&'()*+,:;=@[]" 不含双引号 "，虽有特殊含义，但在 path 部分，编码与否，都可以正常访问;
# 另一些，如 双引号 "，空格 " "，汉字等必须编码;
#
# 建议 url 的 path 中尽量不含上述部分，建议 url 中尽量不含上述部分。
#
# 参考
# https://www.wikiwand.com/zh-cn/%E7%99%BE%E5%88%86%E5%8F%B7%E7%BC%96%E7%A0%81
# https://www.wikiwand.com/en/Percent-encoding
# https://www.wikiwand.com/de/URL-Encoding
    '''

def signt_help():
    print
    print "./signt.py time <key> <url> <t,eg: 3600>"
    print "./signt.py deadline <key> <url> <deadline>"
    print "./signt.py check <key> <signed_url>"
    print "./signt.py show <t, eg: 55bb9b80>"
    print "./signt.py genkey"
    print "\n"
    print '''
# example:

# url = "http://xxx.yyy.com/DIR1/中文/vodfile.mp4?sfdf=dfe"
# key = 12345678
# 过期时间点: Sat Aug  1 00:00:00 2015  ==> 1438358400

# 执行: signt.py deadline 12345678  http://xxx.yyy.com/DIR1/中文/vodfile.mp4?sfdf=dfe 1438358400
# 签名 url 为: http://xxx.yyy.com/DIR1/%E4%B8%AD%E6%96%87/vodfile.mp4?sfdf=dfe&sign=6356bca0d2aecf7211003e468861f5ea&t=55bb9b80

# 执行: signt.py check 12345678 "http://xxx.yyy.com/DIR1/%E4%B8%AD%E6%96%87/vodfile.mp4?sfdf=dfe&sign=6356bca0d2aecf7211003e468861f5ea&t=55bb9b80"
# 显示: True
    '''
    print "\n"
    printurl_encode_help()

def sign_time(key, url, rang):
    print("range: " + str(rang))
    deadline = to_deadline(rang)
    sign_deadline(key, url, deadline)

def sign_deadline(key, url, deadline):
    print("\nkey: " + key)
    print("\nurl: " + url)
    print("\ndeadline: " + t16(deadline) + ", " + str(deadline) + ", " + time.ctime(deadline))
    print
    t = t16(deadline)
    signed_url = sign_url(key, t, url)
    print "\nsigned_url:"
    print signed_url
```

```
    print signed_url
    print

# signed_url 是正确 url_encode 编码后签出的 url
# 见 url_encode 方法注释
def sign_check(key, signed_url):
    print "\n 要求: 待检测的 url 是正确 url_encode 编码后签出的 url"
    printurl_encode_help()
    u = urlparse(signed_url)
    t = parse_qs(u.query)["t"][0]
    sign_s = summd5(key + u.path + t).lower()
    print
    print("deadline: " + str(int(t, 16)) + " , " + time.ctime(int(t, 16)))
    print(sign_s)
    print(parse_qs(u.query)["sign"][0] == sign_s)

def show_t(t):
    i_t = int(t, 16)
    s_t = time.ctime(i_t)
    print(t + " : " + str(i_t) + " : " + s_t)

# 仅用于测试
def gen_key():
    print base64.urlsafe_b64encode(str(uuid.uuid4()))[:40].lower()

try:
    type = sys.argv[1]

    if type == "time":
        sign_time(sys.argv[2], sys.argv[3], int(sys.argv[4]))
    elif type == "deadline":
        sign_deadline(sys.argv[2], sys.argv[3], int(sys.argv[4]))
    elif type == "check":
        sign_check(sys.argv[2], sys.argv[3])
    elif type == "show":
        show_t(sys.argv[2])
    elif type == "genkey":
        gen_key()
    else:
        signt_help()
except Exception as e:
    print traceback.format_exc()
    signt_help()
```