میان‌افزار Form Post

محتویات

میان‌افزار Form Post
فرمت Form Post
امضای HMAC-SHA1 برای form POST
اضافه کردن فایل با prefix
کد تولیدکننده‌ی html با python
بررسی کد

میان‌افزار Form Post

به کمک این میان‌افزار، کاربر می‌تواند اشیاء را مستقیماً از طریق مرورگر و از طریق فرم‌های موجود در آن، به سامانه‌ی ذخیره‌سازی اشیاء ارسال کند. این میان‌افزار از کلیدهای مخفی (secret key) حساب یا کانتینر برای تولید امضای رمزنگاری برای درخواست استفاده می‌کند. این به این معناست که کاربر، نیازی به ارسال توکن احراز هویت در هدر X-Auth-Token برای انجام درخواست ندارد. میان‌افزار Form Post از همان کلیدهای مخفی استفاده می‌کند که میان‌افزار tempurl، استفاده می‌کند. توضیحات در رابطه با این کلیدها در بخش قبلی به صورت کامل آمده‌است.

فرمت Form Post

برای آپلود اشیا به کلاستر به صورت مستقیم و بدون نیاز به احراز هویت، کاربر می‌تواند از درخواست Form Post استفاده کند.

فرمت یک فرم به صورت زیر خواهد بود:

```
<form action="SWIFT_URL"
    method="POST"
    enctype="multipart/form-data">
    <input type="hidden" name="redirect" value="REDIRECT_URL"/>
    <input type="hidden" name="max_file_size" value="BYTES"/>
    <input type="hidden" name="max_file_count" value="COUNT"/>
    <input type="hidden" name="expires" value="UNIX_TIMESTAMP"/>
    <input type="hidden" name="signature" value="HMAC"/>
    <input type="file" name="FILE_NAME"/>
    <br/>
    <input type="submit"/>
</form>
```

در ادامه به توضیح بخش‌های مختلف این درخواست html پرداخته می‌شود.

"action="SWIFT_URL

در این مولفه، Url به صورت کامل تنظیم شود که اشیاء باید به کجا بارگذاری شوند. نام فایل‌های بارگذاری‌شده به URL مشخص شده SWIFT_URL الحاق می‌شوند. بنابراین، کاربر می‌تواند مستقیماً به ریشه‌ی یک کانتینر با یک URL مانند این بارگذاری کند:

```
https://swift-cluster.example.com/v1/my_account/container/
```

هم‌چنین کاربر به صورت اختیاری می‌تواند پیشوندی برای اشیا اتخاذ کند و آپلودهای به مسیر زیر همراه با پیشوند ارسال کند:

```
https://swift-cluster.example.com/v1/my_account/container/OBJECT_PREFIX
```

”method=”POST

متد ارسالی در فرم همواره باید POST باشد. شرطی در میان‌افزار وجود دارد که فقط متد ارسالی پست را بررسی می‌کند و از نام میان‌افزار هم هم‌چین چیزی بر می‌آید.

”enctype=”multipart/form-data

این مقدار نیز همواره باید multipart/form-data باشد.

”name=”redirect” value=”REDIRECT_URL

این مقدار مشخص می‌کند که بعد از ارسال فرم، کاربر به چه صفحه‌ای هدایت می‌شود. URL حاوی کوئری پارامترهای status و message است که کد وضعیت HTTP برای بارگذاری و پیام خطا اختیاری را مشخص می‌کنند. کد وضعیت 2nn موفقیت را نشان می‌دهد.

REDIRECT_URL می‌تواند رشته‌ای خالی باشد. اگر چنین باشد، هدر پاسخ Location پر نمی‌شود.

”name=”max_file_size” value=”BYTES

این مقدار اجباری است. مشخص می‌کند که بیشینه‌ی سایزی که توسط کاربر می‌تواند بارگذاری شود چه اندازه است.

”name=”max_file_count” value= “COUNT

این مقدار اجباری است. مشخص می‌کند که بیشینه‌ی تعداد فایلی که می‌تواند از طریق Form Post بارگذاری شود، چه تعدادی است.

”name=”expires” value=”UNIX_TIMESTAMP

مقدار UNIX timestamp که مشخص می‌کند زمانی که فرم تا پیش از آن می‌تواند بارگذاری شود، چه زمانی در این فرمت است. بعد از آن فرم بی اعتبار خواهد بود.

”name=”signature” value=”HMAC

مقدار امضا با فرمت HMAC-SHA1 از فرم.

”type=”file” name=”FILE_NAME

نام فایل بارگذاری شده. می‌تواند به تعداد فایل‌های بارگذاری شده باشد.

مشخصات فایل باید بعد از سایر مشخصات ظاهر شوند تا به درستی پردازش شوند.

اگر مشخصات اصلی، بعد از مشخصات فایل ظاهر شوند، با درخواست ارسال نمی‌شوند، زیرا تمام مشخصات موجود در فایل نمی‌توانند به صورت درست پارس شوند مگر اینکه تمام فایل به حافظه خوانده شود؛ اما سرور حافظه کافی برای پاسخ‌گویی به این درخواست‌ها ندارد. پس مشخصاتی که پس از صفات فایل ظاهر می‌شوند، نادیده گرفته می‌شوند.

در صورت تمایل، اگر می‌خواهید فایل‌های بارگذاری‌شده به صورت موقت باشند، می‌توانید با افزودن یکی از مشخصات x-delete-at یا x-delete-after به عنوان ورودی فرم، تنظیم کنید:

- x-delete-at: این مشخصه باید به عنوان یکی از ورودی‌های فرم اضافه شود و باید تاریخ و زمانی مشخص کند که فایل باید از سیستم حذف شود.
- x-delete-after: این مشخصه همچنین باید به عنوان یکی از ورودی‌های فرم اضافه شود و باید مقدار زمانی (به عنوان مثال تعداد ثانیه) را مشخص کند که باید پس از بارگذاری فایل، فایل حذف شود.

با اضافه کردن یکی از این صفات، می‌توانید فایل‌های بارگذاری‌شده را به صورت موقت تنظیم کنید و از اتلاف فضای ذخیره‌سازی جلوگیری کنید.

```
<input type="hidden" name="x_delete_at" value="<unix-timestamp>" />
<input type="hidden" name="x_delete_after" value="<seconds>" />
```

”type= “submit

این نوع باید برابر submit باشد.

امضای HMAC-SHA1 برای form POST

میان‌افزار form POST از امضای رمزنگاری HMAC-SHA1 استفاده می‌کند. این امضا شامل این عناصر از فرم می‌شود:

۱. Path. مسیر ابتدا با /v1/ شروع شده و شامل نام کانتینر و به صورت اختیاری پیشوند شیء است. در مثال، مسیر به صورت /v1/my_account/container/object_prefix می‌باشد. در این مرحله مسیر نباید URL-encode شود.

۲. Redirect URL. اگر هیچ URL تغییر مسیری وجود ندارد، از رشته‌ی خالی استفاده شود.

۳. بیشینه‌ی اندازه‌ی فایل. در مثال، حداکثر اندازه فایل برابر با ۱۰۴۸۵۷۶۰۰ بایت (104857600 بایت) است.

۴. حداکثر تعداد اشیاء برای بارگذاری. در مثال، حداکثر تعداد فایل‌ها برابر با ۱۰ است.

۵. زمان انقضا. در مثال، زمان انقضا به صورت ۶۰۰ ثانیه در آینده تنظیم شده است. دقت شود که عدد نهایی expire در امضا کلیدی و مهم است، نه زمانی که برای انقضا در آینده تعیین می‌شود.

۶. کلید مخفی. به عنوان مقدار هدر X-Account-Meta-Temp-URL-Key برای حساب‌ها یا مقدار هدر X-Container-Meta-Temp-URL-Key برای کانتینرها تنظیم می‌شود. توضیحات کامل در مورد کلیدها در بخش tempurl آمده است.

مثال:

```python
import hmac
from hashlib import sha1
from time import time
path = '/v1/my_account/container/object_prefix'
redirect = 'https://myserver.com/some-page'
max_file_size = 104857600
max_file_count = 10
expires = int(time() + 600)
key = 'MYKEY'
hmac_body = '%s\n%s\n%s\n%s\n%s' % (path, redirect,
max_file_size, max_file_count, expires)
signature = hmac.new(key, hmac_body, sha1).hexdigest()
```

اضافه کردن فایل با prefix

برای اضافه کردن فایل با prefix، کافی امضایی که ساخته می‌شود همراه با prefix باشد. علاوه بر این مسیری که در قسمت action_url می‌آید هم باید همراه با prefix باشد. مثلا نمونه‌ی html زیر، فرمی است که با prefix اشیا را بارگذاری می‌کند.

```html
<form
  action="http://localhost:8080/v1/AUTH_admin/c1/pre/"
  method="POST"
  enctype="multipart/form-data"
>
  <!-- Hidden fields -->
  <input type="hidden" name="redirect" value="https://burna.com" />
  <input type="hidden" name="max_file_size" value="104857600" />
  <input type="hidden" name="max_file_count" value="10" />
  <input type="hidden" name="expires" value="2691216900" />
  <input
    type="hidden"
    name="signature"
    value="6f5edbc6752b702361a343d9761c4af30e911c5a"
  />

  <!-- File input fields -->
  <input type="file" name="file1" /><br />
  <input type="file" name="file2" /><br />
  <input type="file" name="file3" /><br />

  <!-- Submit button -->
```

```
  <input type="submit" />
</form>
```

کد تولیدکننده‌ی html با python

برای تولید کد html، اسکریپت پایتون ۲ زیر نوشته شده است که این محتوا را به صورت خودکار با توجه به متغیرهایی که در آن آمده است آماده کرده و فایل html را خروجی می‌دهد.

```
import hmac
from hashlib import sha1
from time import time

# Parameters determined by developer
swift_base_url = 'http://localhost:8080'
path = '/v1/AUTH_admin/c1/pre/'
filenames_to_upload = 'file1 file2 file3'.split()
redirect = 'https://varzesh3.com'
max_file_size = 104857600
max_file_count = 10
timeout = 6000
key = 'secret123'
expires = 2691216900

hmac_body = '%s\n%s\n%s\n%s\n%s' % (path, redirect,
                                    max_file_size, max_file_count, expires)

signature = hmac.new(key.encode(), hmac_body.encode(), sha1).hexdigest()

swift_url = "%s%s" % (swift_base_url, path)

form_header = """
<form action="%(swift_url)s" method="POST" enctype="multipart/form-data">
  <!-- Hidden fields -->
  <input type="hidden" name="redirect" value="%(redirect)s" />
  <input type="hidden" name="max_file_size" value="%(max_file_size)s" />
  <input type="hidden" name="max_file_count" value="%(max_file_count)s" />
  <input type="hidden" name="expires" value="%(expires)s" />
  <input type="hidden" name="signature" value="%(signature)s" />

  <!-- File input fields -->
  %(file_inputs)s

  <!-- Submit button -->
  <input type="submit" />
</form>
"""

file_inputs = ''.join(['  <input type="file" name="%s" /><br />\n' % filename
                       for filename in filenames_to_upload])

output_html = form_header % {
    'swift_url': swift_url,
    'redirect': redirect,
    'max_file_size': max_file_size,
    'max_file_count': max_file_count,
    'expires': expires,
    'signature': signature,
    'file_inputs': file_inputs,
}

# Write to the file
with open('base.html', 'w') as file:
    file.write(output_html)
```

بررسی کد

کد این میان‌افزار شامل چند بخش اصلی می‌باشد که در زیر به تفکیک توضیح داده خواهند شد.

تابع _get_keys

این تابع به دستگاه‌های `X-[Account|Container]-Meta-Temp-URL-Key[-2]` مرتبط با حساب یا کانتینر برمی‌گردد. اگر هیچ کلیدی تنظیم نشده باشد، یک لیست خالی برگشت داده می‌شود.

در ابتدا، مسیر درخواست (PATH_INFO) به اجزای خود تقسیم می‌شود تا اطلاعات ضروری مانند نسخه API، حساب و کانتینر به دست آید. اگر اجزای مورد نیاز موجود نباشد (مانند اندازه کمتر از 4، عنصر اول خالی، نسخه API نامعتبر و یا حساب و کانتینر ناپردازش شده)، لیست خالی برگشت داده می‌شود.

سپس اطلاعات حساب و کانتینر با فراخوانی توابع `get_account_info` و `get_container_info` از متغیرهای محیطی (env) گرفته می‌شوند. کلیدهای مرتبط با Temporary URL از اطلاعات متا (meta) حساب و کانتینر استخراج می‌شوند.

در نهایت، لیست کلیدهای حساب و کانتینر با یکدیگر ترکیب می‌شوند و به عنوان نتیجه بازگردانده می‌شوند. این تابع به ازای هر حساب و کانتینر می‌تواند تا 4 کلید را بازگرداند.

تابع _translate_form

این تابع ترجمه‌ی داده‌های فرم به زیردرخواست‌ها را انجام می‌دهد و پاسخی ارسال می‌کند. ورودی تابع شامل متغیرهای `env` (محیط WSGI) و `boundary` (مرز نوع MIME) می‌باشد.

این تابع ابتدا کلیدهای مرتبط با Temporary URL را با فراخوانی تابع `_get_keys(env)` بدست می‌آورد. سپس با توجه به نوع و مشخصات فایل‌ها و محتواهای دیگری که در فرم ارسال شده‌اند، ترجمه می‌شوند و به شکل زیردرخواست‌ها به سمت مخازن می‌فرستند.

در این فرآیند، تعداد فایل‌ها شمرده شده و اگر این تعداد از تنظیمات مجاز (به عنوان مثال ماکسیمم تعداد فایل‌ها) بیشتر باشد، بازخورد مناسبی ایجاد می‌شود. سپس اطلاعات فایل‌های فرستاده شده، مثل نام، نوع محتوا و محدودیت‌های دیگر با استفاده از تابعهای پارس و بررسی می‌شوند.

نتیجه نهایی شامل وضعیت پاسخ (status)، هدرها (headers) و محتوای بدنه (body) به صورت یک دیکشنری بازگشت داده می‌شود. اگر عملیات ترجمه و ترجمه زیردرخواست‌ها با موفقیت انجام شود، اطلاعات مربوط به ترجمه و ارجاع به وضعیت و پیغام در ادامه‌ی عملیات اعلان شده تا کاربر را به صفحه موردنظر هدایت کند.

تابع _perform_subrequest

این تابع زیردرخواست را انجام می‌دهد و پاسخ آن را بازمی‌گرداند. ورودی‌های تابع شامل `orig_env` (محیط WSGI اصلی)، `attributes` (مشخصات فرم)، `fp` (شیء شبه فایل حاوی بدنه درخواست) و `keys` (کلیدهای حساب برای اعتبارسنجی امضا) می‌شود.

در ابتدا، تعداد کلیدهای مرتبط با Temporary URL با فراخوانی تابع `_get_keys(env)` بررسی می‌شود. سپس حداکثر اندازه فایلی که می‌تواند ارسال شود از مشخصات فرم دریافت می‌شود. سپس یک محیط WSGI جدید برای زیردرخواست ایجاد می‌شود.

در زیردرخواست، ورودی `wsgi.input` به عنوان یک شیء شبه فایل کاهش می‌یابد و تغییرات لازم به متغیرهای محیطی اعمال می‌شود، مانند تغییر PATH_INFO برای مسیر فایل. همچنین، اگر مشخصاتی مانند زمان انقضا (expires) یا محدودیت‌های دیگر مشخص شده باشند، نیز درخواست فراخوانی می‌شود.

در ادامه، امضا روی بدنه درخواست ایجاد می‌شود و با استفاده از این امضا و کلیدهای اعتبارسنجی، اعتبارسنجی امضا انجام می‌شود. اگر اعتبارسنجی با موفقیت انجام شود، یک زیردرخواست انجام می‌شود و پاسخ آن بازمی‌گردانده می‌شود.

نتیجه نهایی شامل خط وضعیت (status)، لیست هدرها (headers) و محتوای بدنه (body) پاسخ به عنوان یک دیکشنری بازگشت داده می‌شود. اگر امضا معتبر نباشد یا اندازه فایل با اندازه مشخص شده بیشتر شود، استثناهای مناسب ایجاد می‌شوند و اعلان شده تا پردازش از دست برود.

تابع _call

این تابع بخش اصلی ورودی میان‌افزار است. وظیفه اصلی این تابع بررسی درخواست و واکنش مناسب به آن است.

زمانی که متغیر محیطی `REQUEST_METHOD` برابر با "POST" باشد، تابع به بررسی نوع محتوای درخواست و تجزیه و تحلیل مشخصات دیسپوزیشن می‌پردازد. اگر نوع محتوای درخواست "multipart/form-data" باشد و یک مشخصه با نام "boundary" در مشخصات دیسپوزیشن وجود داشته باشد، تابع `_translate_form` برای ترجمه داده‌های فرم و ایجاد

زیردرخواست‌ها صدا زده می‌شود. در نهایت، واکنش مناسب با توجه به نتایج برگشتی از `translate_form_` تولید و به عنوان نتیجه به دستکاری‌کننده `start_response` بازگردانده می‌شود.

در صورت وقوع خطاهای مختلفی از جمله `EOFError`، `FormInvalid`، `MimeInvalid` و `FormUnauthorized`، پاسخ‌های مناسب با کدهای وضعیت و پیام‌های مرتبط تولید و ارسال می‌شوند. اگر درخواست `POST` نباشد، واکنش مناسب از برنامه اصلی WSGI بازگردانده می‌شود.

برگرفته از «https://kateb.burna.ir/w/index.php?title=میان‌افزار_Form_Post&oldid=49268»

مشارکت‌کنندگان: امیر عباس زاده, مبینا حاج ابراهیمی, محمد مهدی ایل بیگی

این صفحه آخرین‌بار در ۲ آذر ۱۴۰۲ ساعت ۰۰:۱۰ ویرایش شده‌است.