

SharkFest'26 Europe

Wireshark with WebBrowser

Megumi Takeshita, Packet Otaku
Ikeriri network service

please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip

Sponsored by: PROFITAP DEEP NETWORK OBSERVABILITY, cPacket, TCP/IP training, endace Record. Respond.

SharkFest'26 Europe BRUSSELS
Wireshark User & Developer Conference

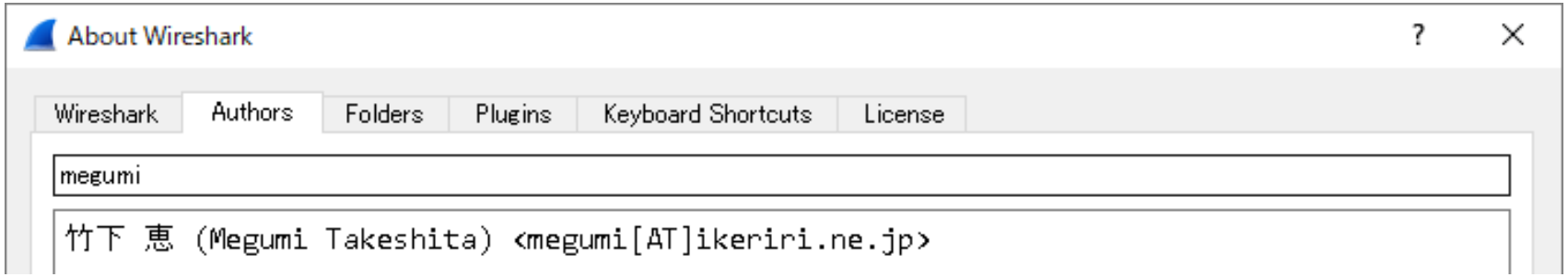
WIRESHARK

Megumi Takeshita, Packet Otaku, ikeriri network service



- Worked SE/IS at BayNetwork, Nortel
- Reseller of CACE technologies in 2008
- Founder, ikeriri network service co., ltd
  Reseller of packet capture / wireless-tools
- Wrote 10+ books about Wireshark in Japanese
- Instruct Wireshark to JSDF etc.
- Lecturer of CHUO University
- One of the contributors to Wireshark
  Translate Wireshark into Japanese





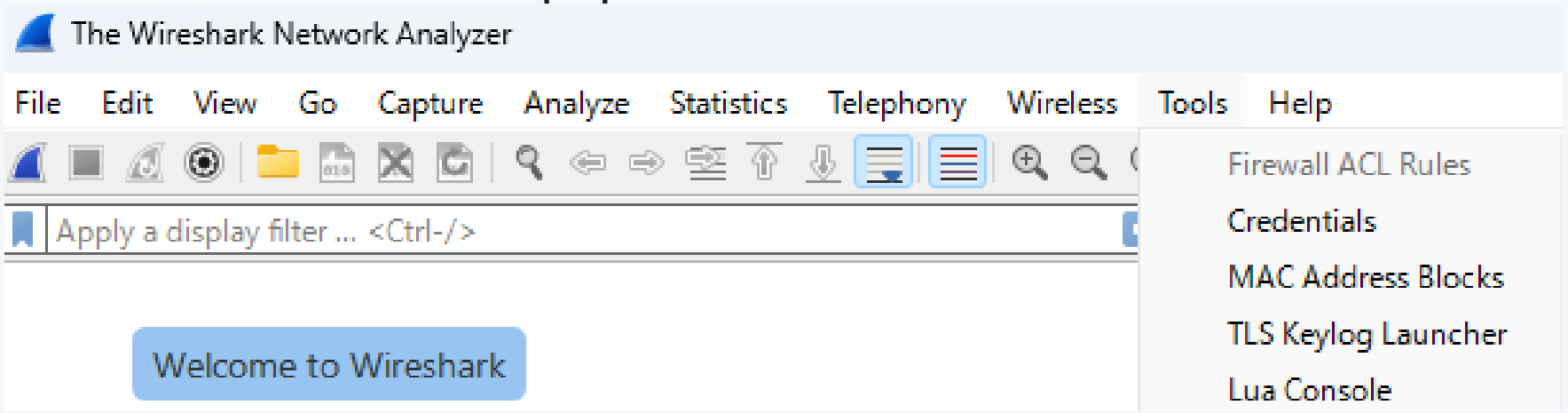
TLS Decryption with WebBrowser

please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip



Google Chrome with Wireshark

- Chrome is the best choice for Wireshark
- SSLKEYLOGFILE is supported by Chrome and developer version of Firefox (No Edge, Safari)
- Check the file path of Chrome, "C:¥Program Files¥Google¥Chrome¥Application¥chrome.exe„
- Tools> TLS Keylog Launcher



SSLKEYLOGFILE (IETF draft)

- SSLKEYLOGFILE is the name of environmental valiables, standardlized by IETF draft https://www.ietf.org/archive/id/draft-thomson-tls-keylogfile-01.txt
- SSLKEYLOGFILE points a path of text file containing TLS session keys. Wireshark can use it to decrypt HTTPS/TLS traffic, as well as QUIC and HTTP2/3 etc.
- SSLKEYLOGFILE uses the format: <Label> <ClientRandom> <Secret> Each line represents a TLS secret generated during a connection. Multiple lines may correspond to the same TLS session.

SSLKEYLOGFILE Sample

- Chrome leaks TLS secrets into SSLKEYLOGFILE

```
<label>                                <ClientRandom> <Secret>
CLIENT_RANDOM                          639f....7c73   df3a....73a7
CLIENT_HANDSHAKE_TRAFFIC_SECRET 5c59.... d647  a056....cb36
SERVER_HANDSHAKE_TRAFFIC_SECRET 5c59.... d647 436e....7e04
CLIENT_TRAFFIC_SECRET_0                5c59....d647  7f9e....d6f2
SERVER_TRAFFIC_SECRET_0                5c59....d647 1550.... 3222
EXPORTER_SECRET                        5c59....d647 ebfe....0dab
```

TLS1.2

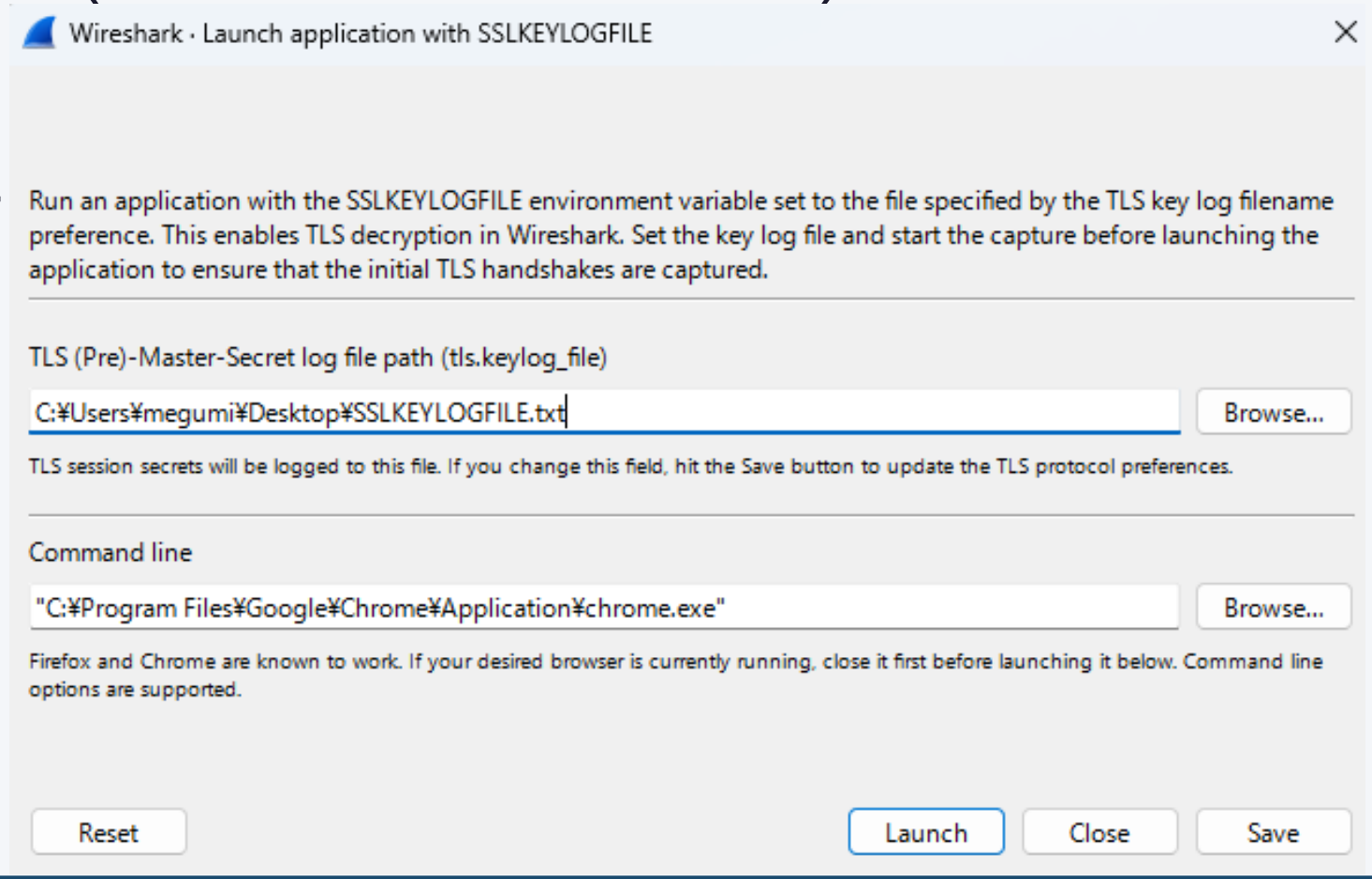
TLS1.3

1 session

- Wireshark decrypts TLS traffic by SSLKEYLOGFILE

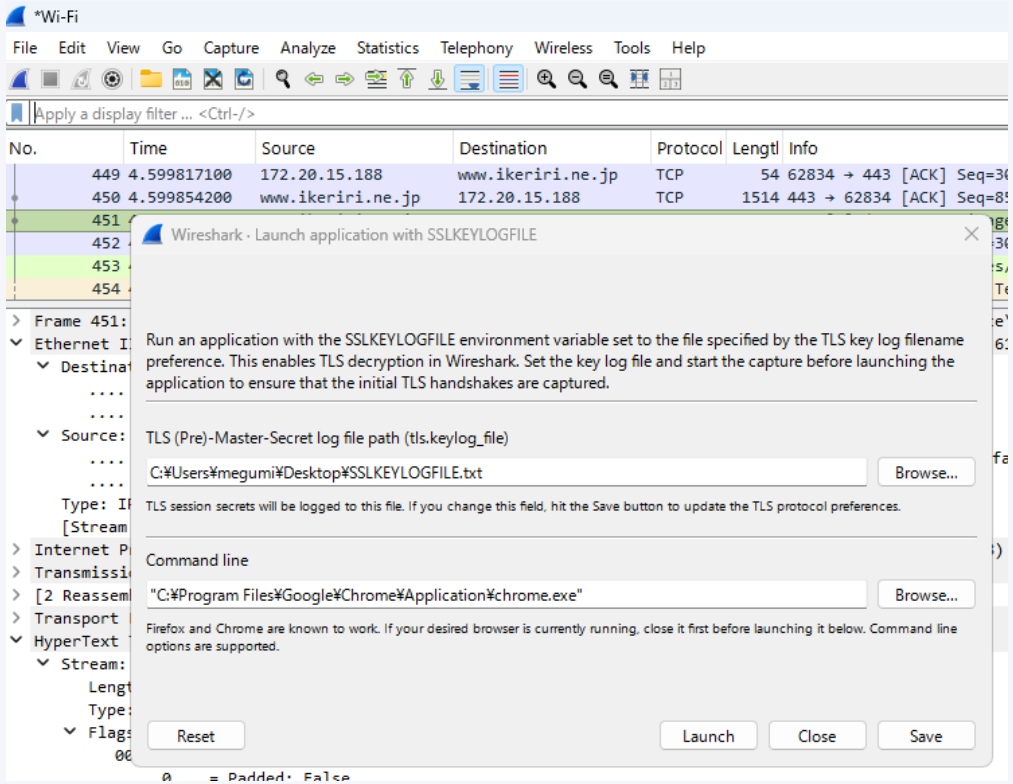
Tools -> TLS Keylog Launcher

- Wireshark can launch Chrome with TLS (Pre)-Master Secret log file (SSLKEYLOGFILE)
- Choose Tools -> TLS Keylog launcher
- Set the paths of SSLKEYLOGFILE and Chrome application
- Save/Launch



Tools -> TLS Keylog Launcher

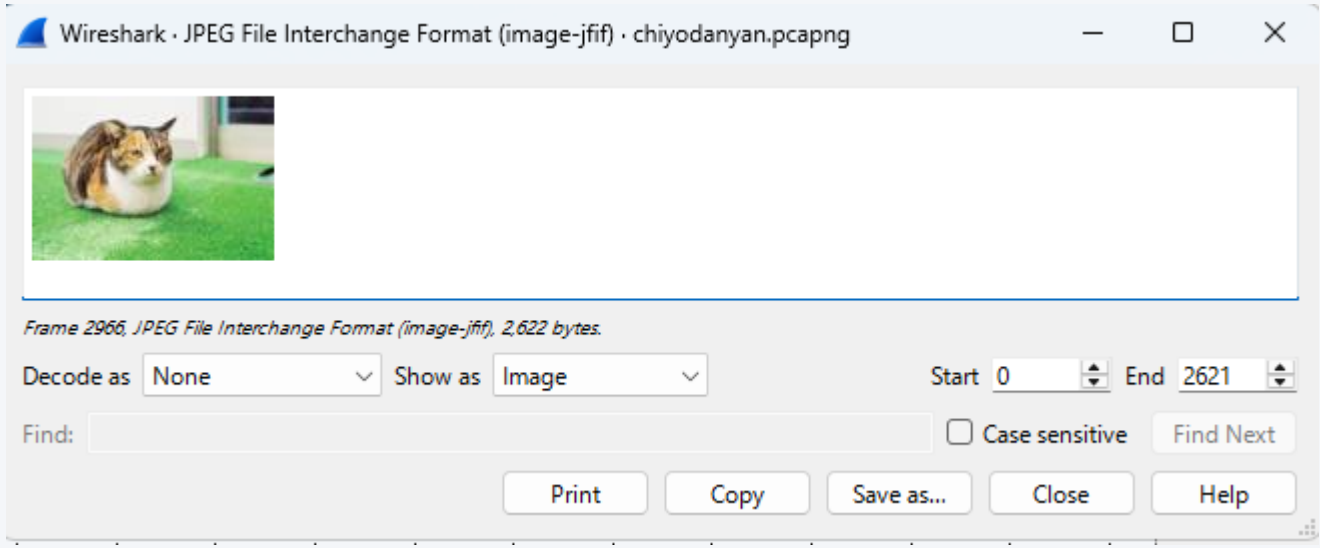
- Start capturing and launch Keylog Launcher
- Ctrl+Alt+Delete to clear Chrome temporary files
  Open https://www.ikeriri.ne.jp/wireshark/en/cheer.html





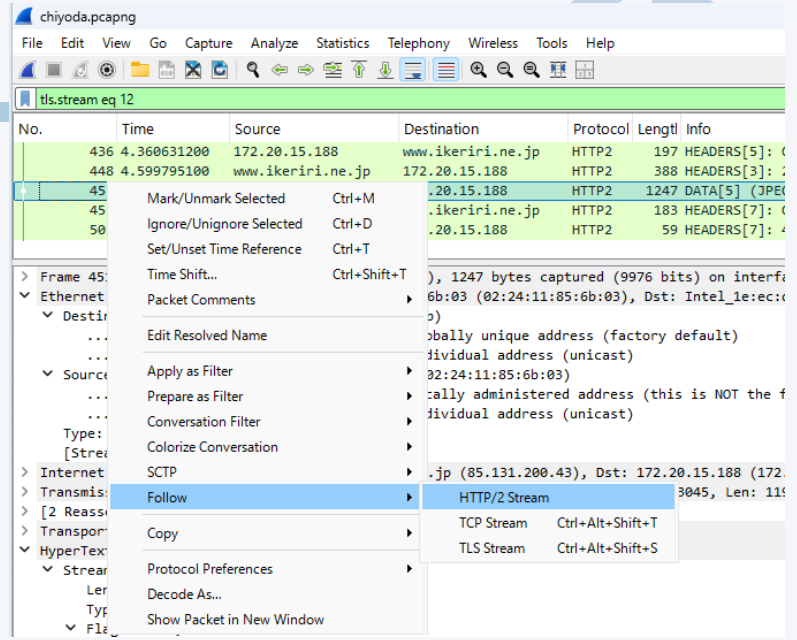
Decrypt TLS traffic

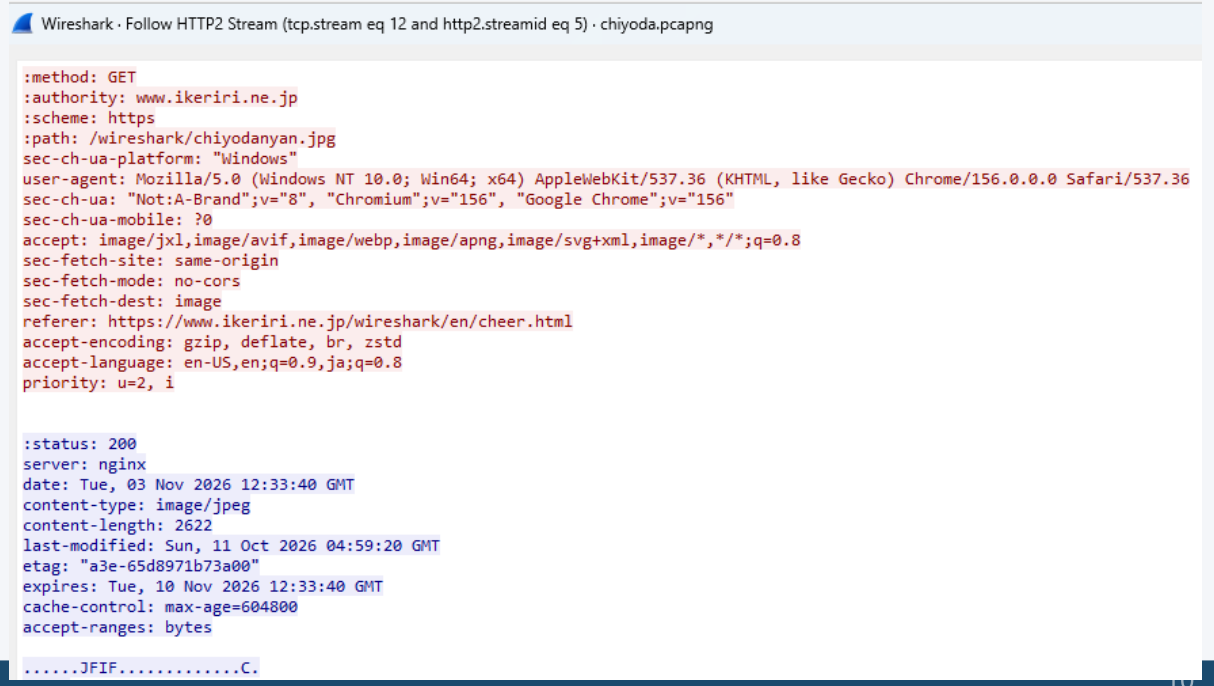
- Stop capturing, save trace as chiyoda.pcapng
- Set Display Filter as image-jfif, pick up a filtered packet and click “JPEG File Interchange Format”, right click and “Show Packet Bytes…”
- We can see choyda nyan.jpg in TLS traffic
- Check encapsulation eth|ipv4|tcp|tls|http2



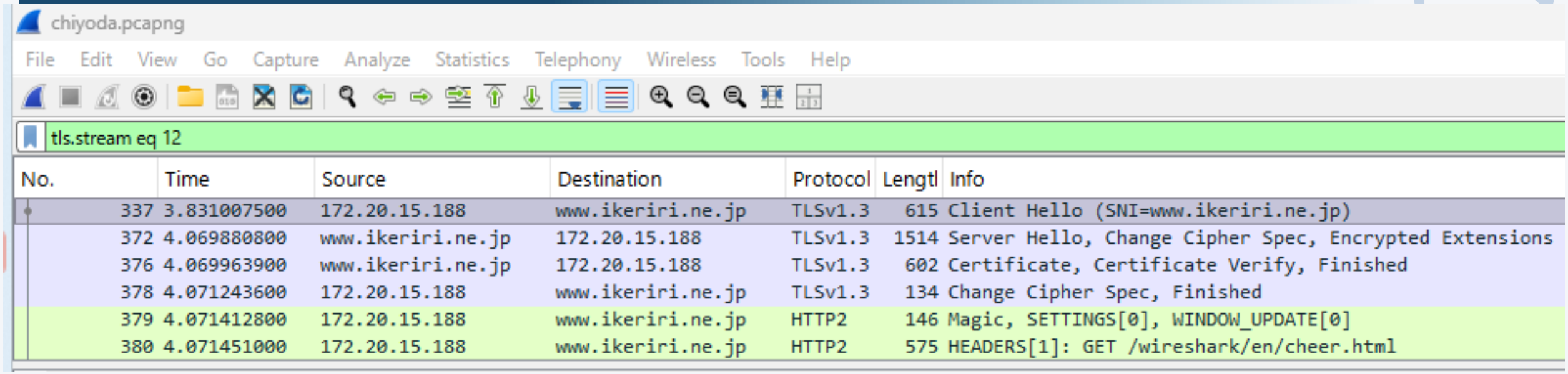
Follow TLS Stream

- Right-click the JPEG packet select Follow -> TLS Stream
- The TLS session is decrypted but we cannot see the header because HTTP2 uses binary headers
- We can check plaintext HTTP/2 request/response decrypted by Wireshark
- Follow -> HTTP/2 Stream also decrypts HTTP/2 binary header and compressions





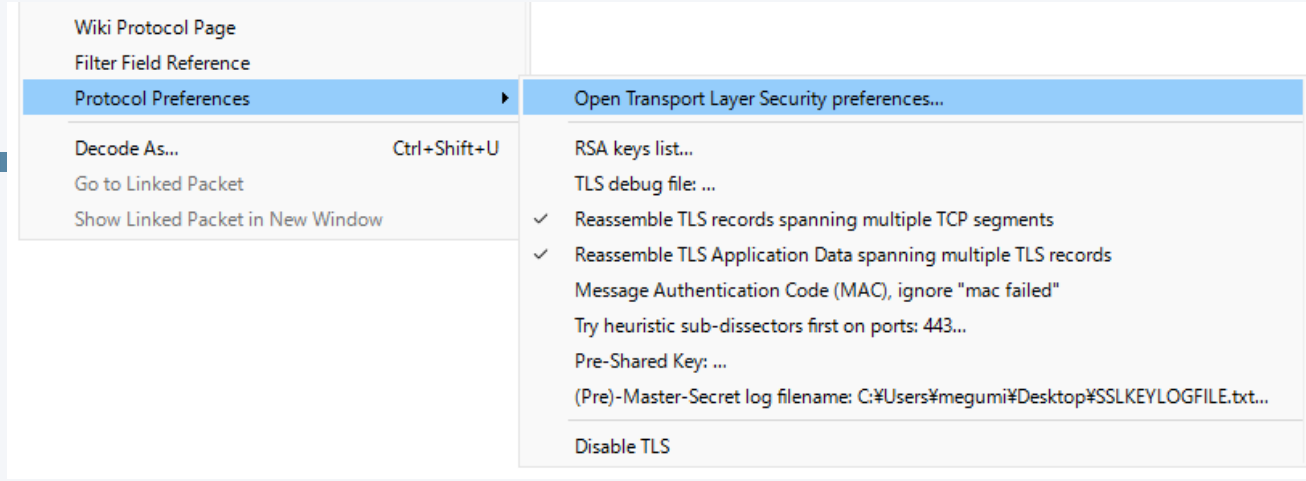
Check the TLS connection process

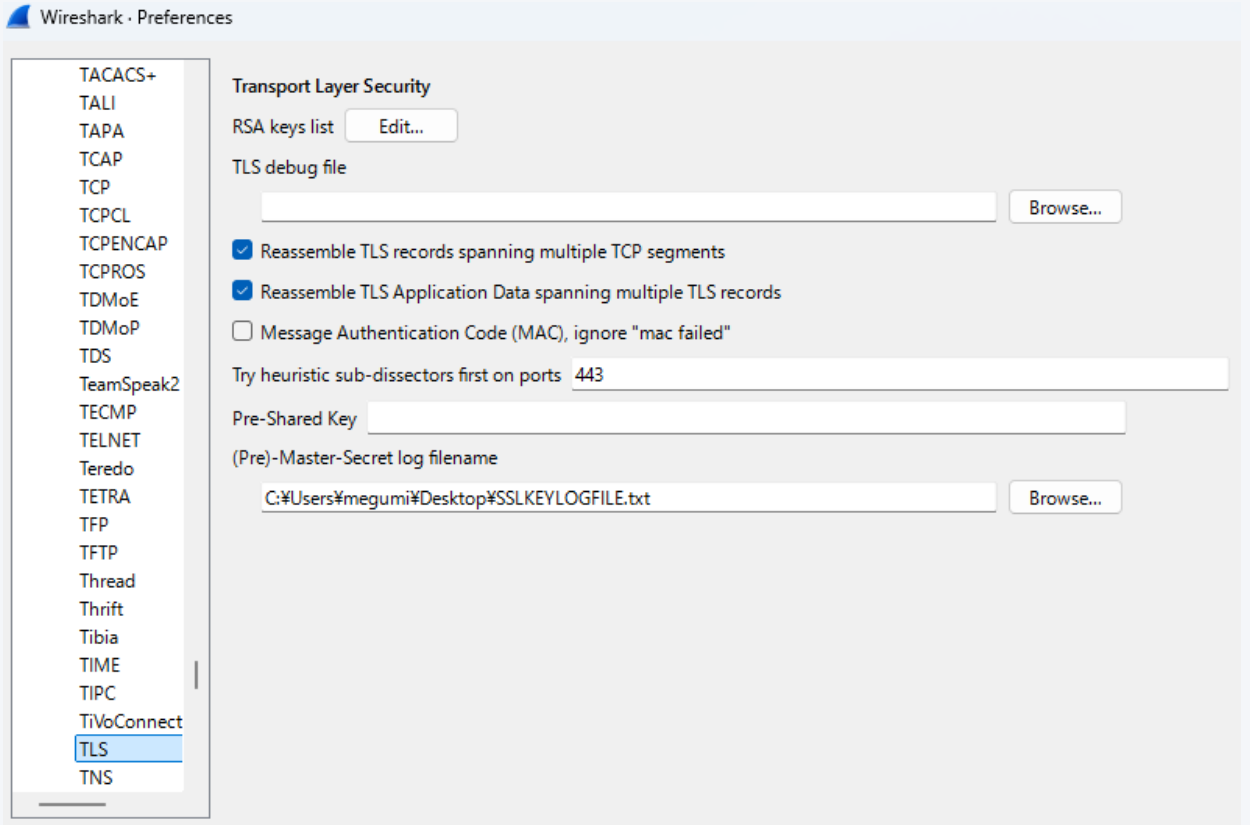


- Follow TLS Stream to pick up the connection; go back to Wireshark, check the first packets.
- We can see Client Hello, Server Hello, Certificate, Client Key Exchange and New Session Ticket.

Check TLS Settings

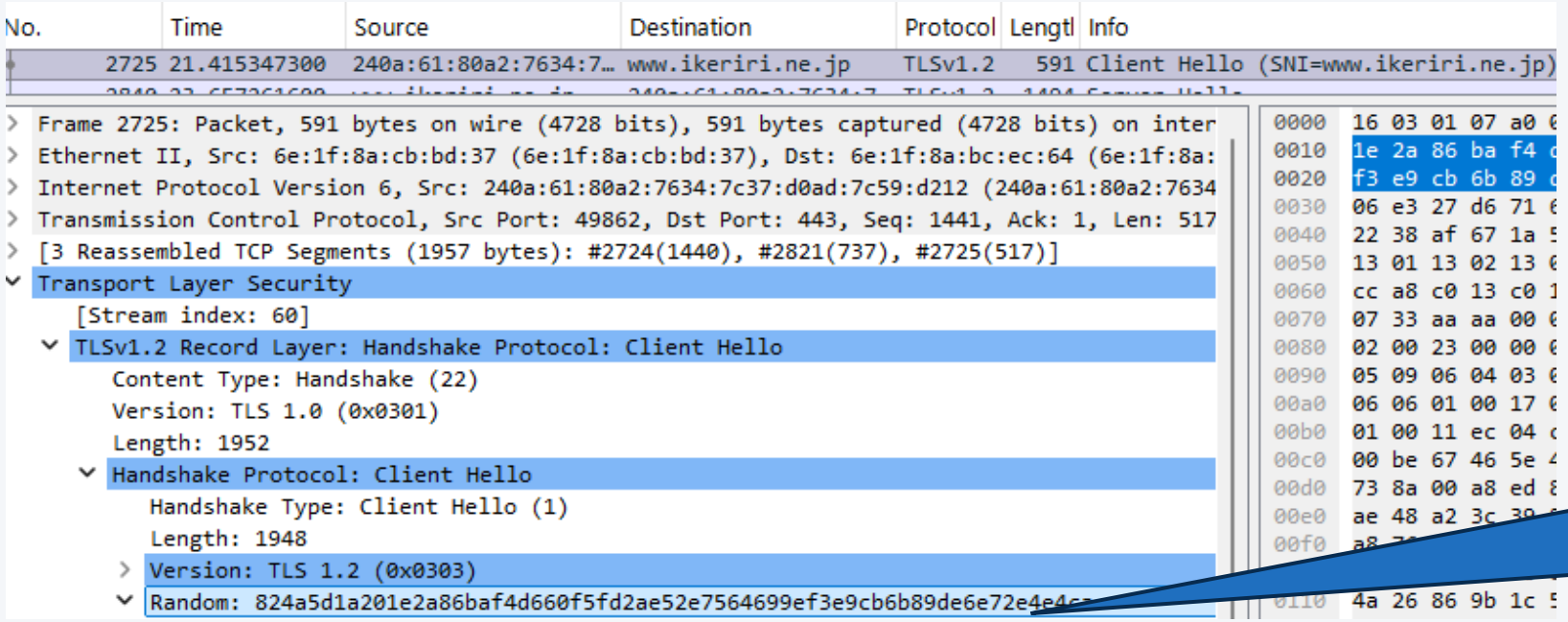
- Choose TLS header, right-click and open Protocol Preferences > Open Transport Layer Security preferences
- (Pre)-Master-Secret log file is set to the same value of TLS KeyLog Launcher





Copy Random value in Client Hello

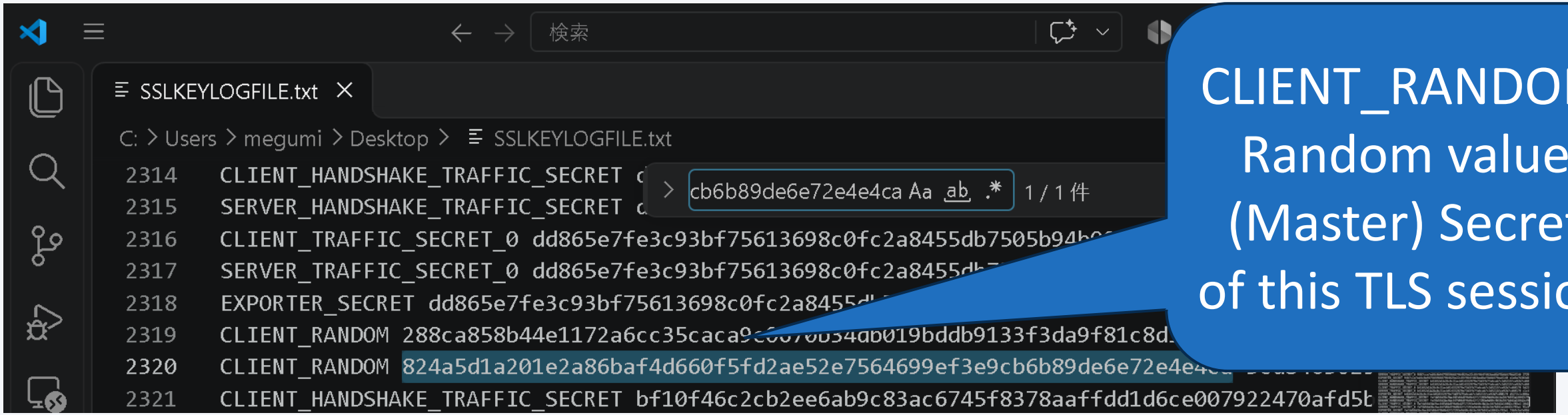
- Choose Client Hello (SNI=www.ikeriri.ne.jp)



- Copy Transport Layer Security -> TLSv1.2 Record Layer: Handshake Protocol: Client Hello -> Handshake Protocol: Client Hello -> Random value

Match Random value as SSLKEYLOGFILE

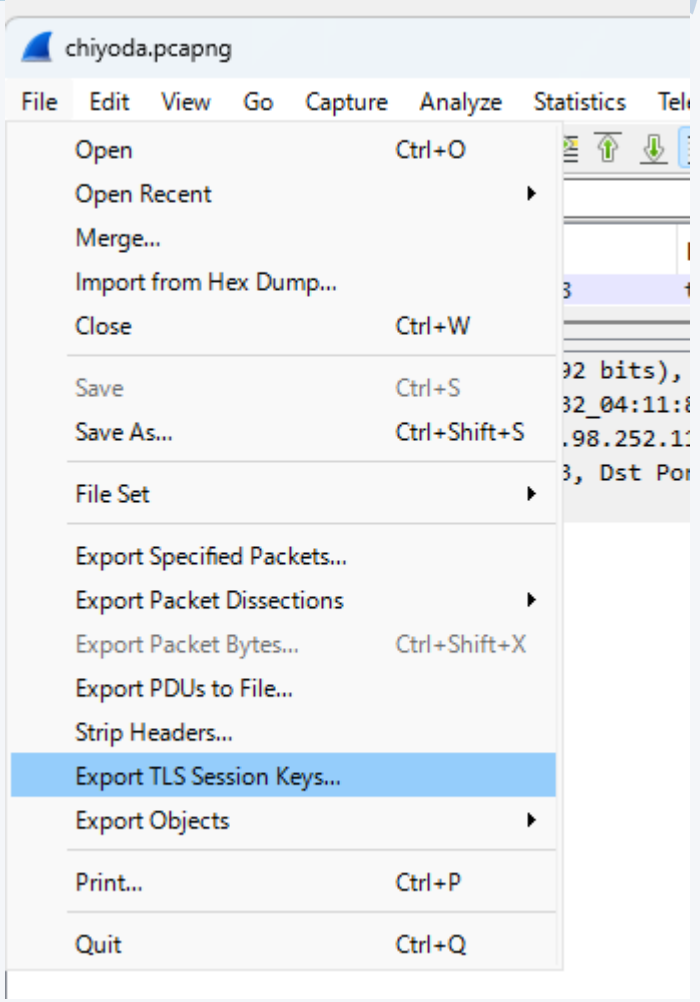
- Open SSLKEYLOGFILE and search Random value of Client Hello -> Hit! CLIENT_RANDOM …

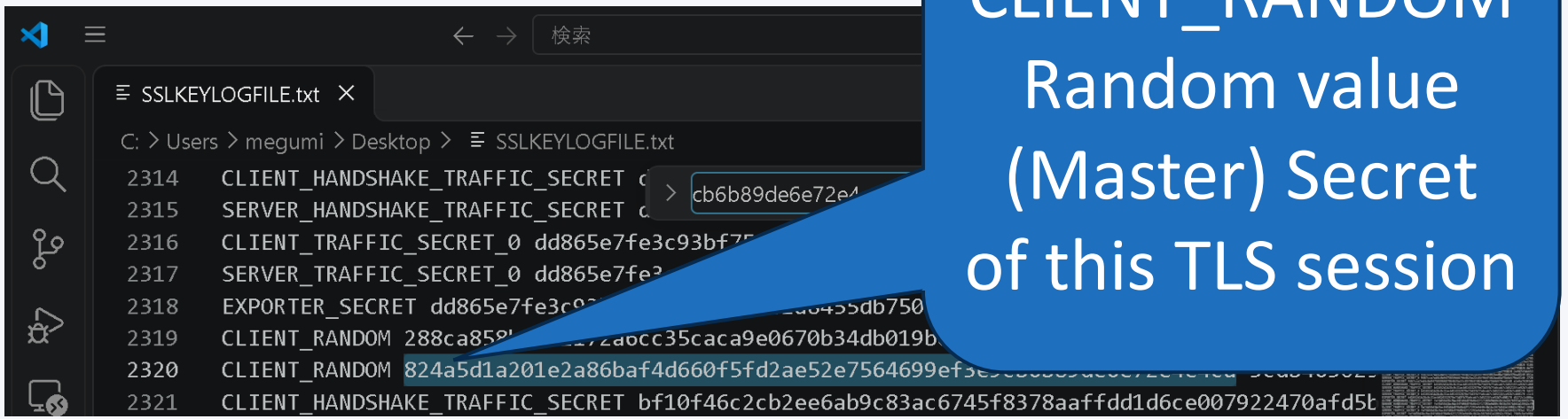


- We can find CLIENT_RANDOM Random value of Client Hello Master Secret (48bytes: TLS1.0,1.1,1.2)

Match Random value as SSLKEYLOGFILE

- You can also export the TLS key File > Export TLS Session Keys...
- Open exported TLS Session key and compare the randam value in Client Hello Packet





CLIENT_RANDOM
Random value
(Master) Secret
of this TLS session

HAR file with Wireshark

please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip



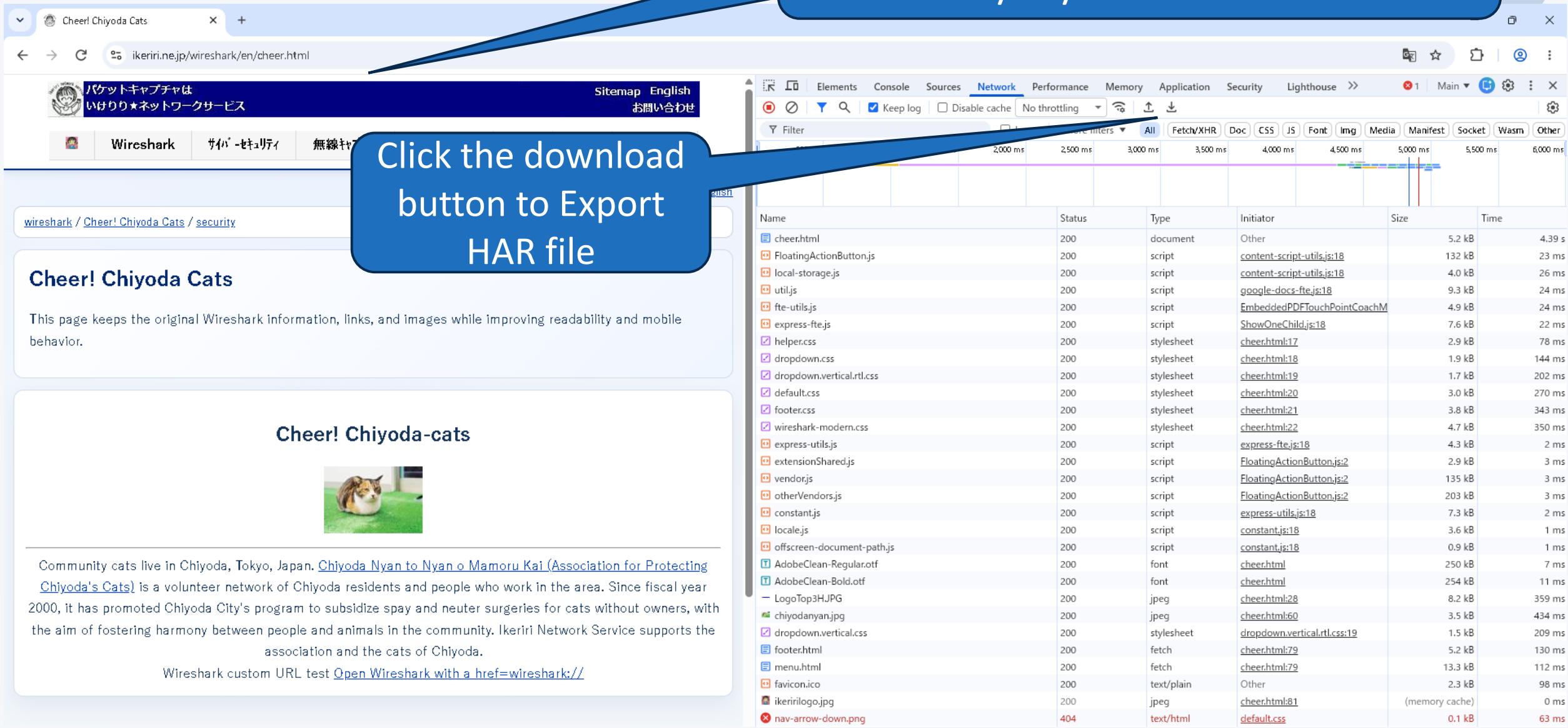
HAR (HTTP Archive)

- HAR file is a JSON file to record HTTP/HTTPS communication between WebBrowser and Server
- HAR file is created by Chrome DevTools, contains URL, HTTP method, HTTP headers, HTTP response, cookies, and timing information.
- Flush History and clear DNS (ipconfig /flushdNS), Open Chrome, push [F12] and choose Network to display Chrome developer tool, and check "keep log" and reload the webpage. https://www.ikeriri.ne.jp/wireshark/en/cheer.html

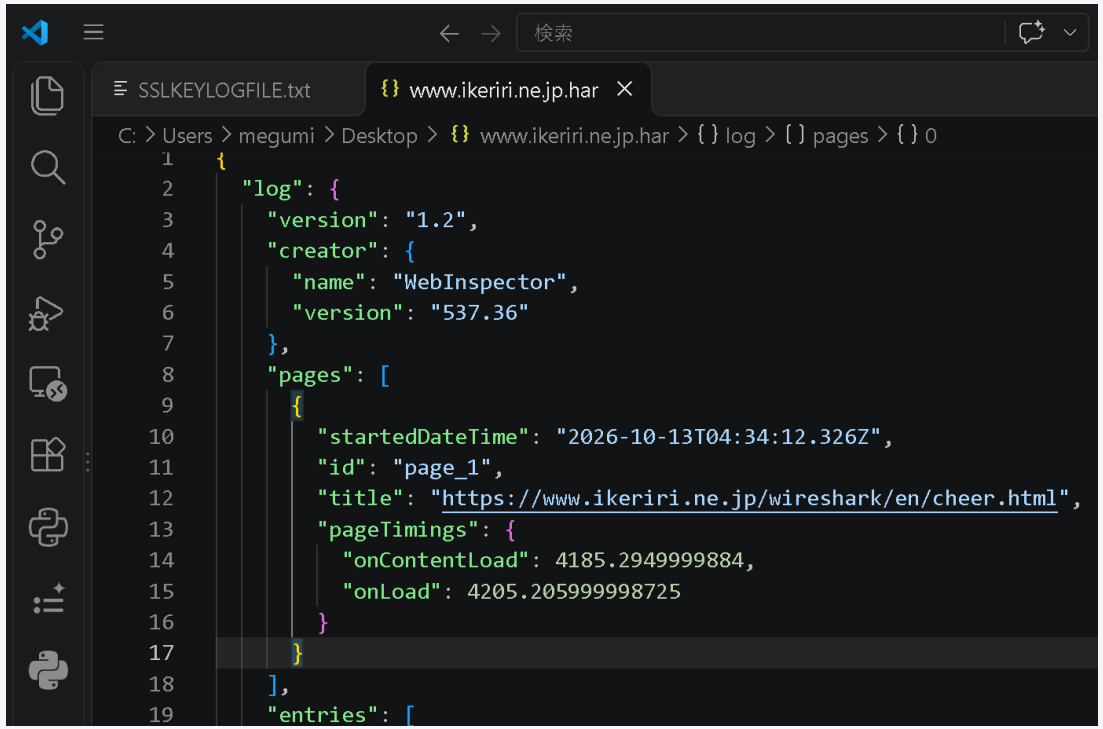
HAR (HTTP Archive)

https://www.ikeriri.ne.jp/wireshark/en/cheer.html

Click the download button to Export HAR file



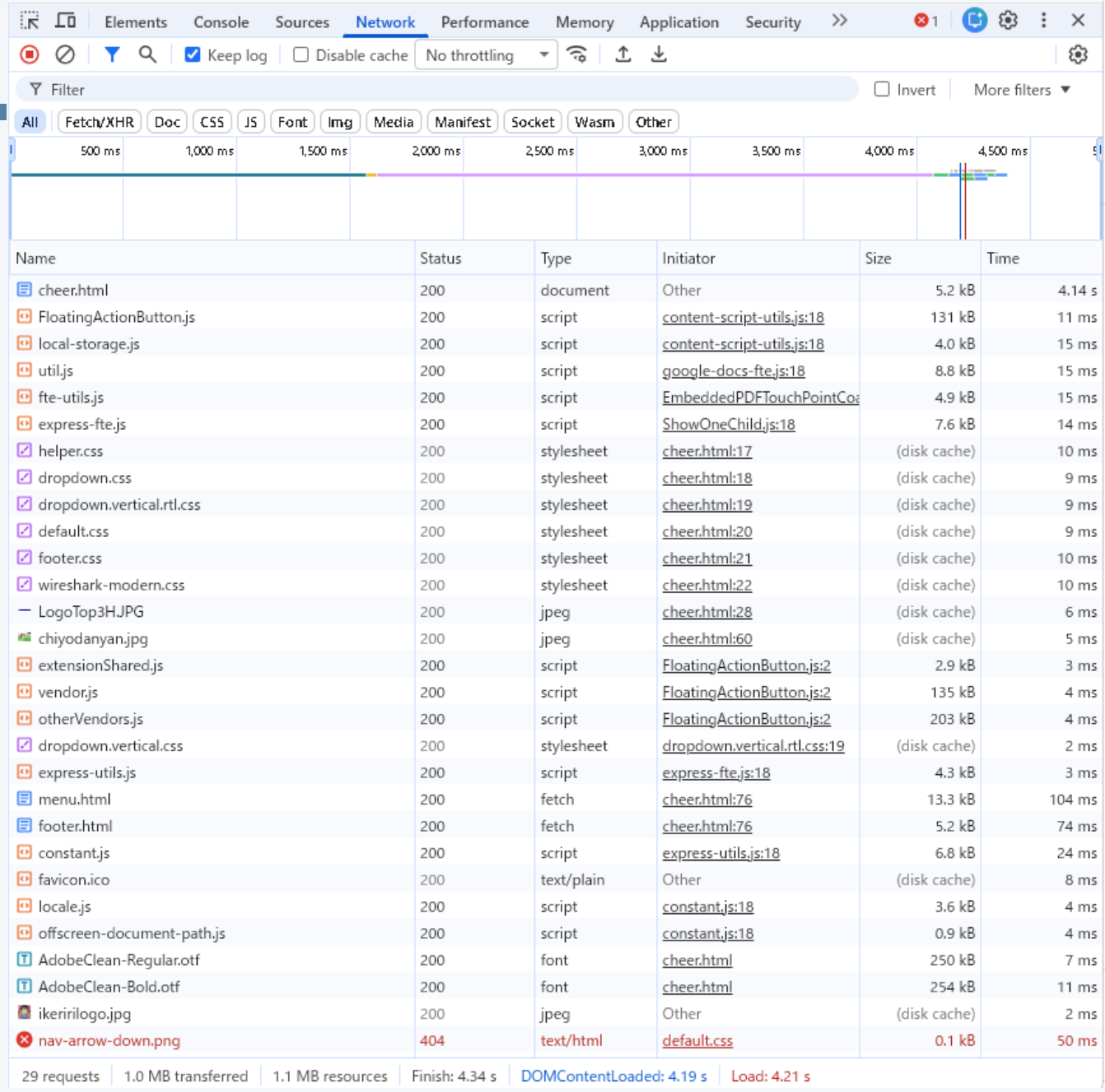
HAR (HTTP Archive)



- Open HAR file with editor
- HAR file contains
  { “request”:{ },
  “response”:{},
  “time”: ,
  “timing”:{} } information
- HAR file contains URL, Headers, status, POST data, response, time and timing (so useful!!)
- HAR file do not need SSLKEYLOGFILE.

Chrome DevTool

- Name, Status, Type, Indicator, Size, Time
- We can find the cause of communication from indicator column (Wireshark cannot)
- Colorized Time graph is easy to understand the latency



Timing information from DevTools



- Click cheer.html and select Timing tab

- We can divide the cheer.html duration 5.7sec into Stalled, DNS lookup, Initial connection, SSL
- Also, we can get Request sent, waiting for server response and Content download time too

Trace(pcap/pcapng) vs HAR file

- Trace file contains all layers and other traffic
- HAR file gives us timing information

Description	Trace file	HAR file
Target	Network side	Application side
Layer	○From layer2 to layer7	△HTTP/HTTPS
TLS decryption	△SSLKEYLOGFILE	○Nothing to prepare
DNS/TLS/HTTP response	○ Generated fields	○ flow diagram
ARP, TCP retransmission and packet lost, window size etc.	○ all layer information	× No record
Web API and WebSocket	△ Difficult for handle	○ Easy to find the clue
Network troubleshoot	○ Perfect	△ Few information

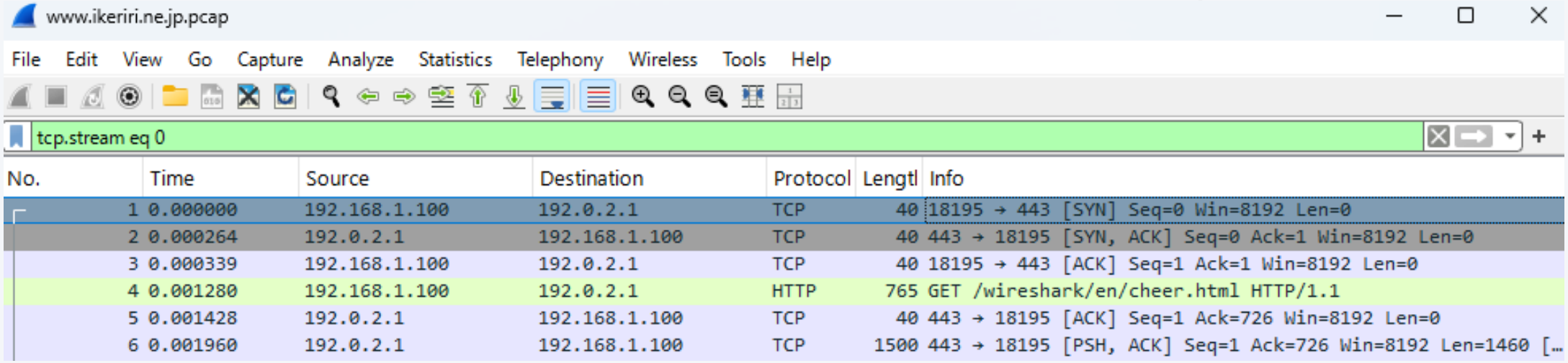
har2cap github.com/wmetcalf/har2pcap

- har2pcap is an open-source project that converts a HAR to a PCAP in Python
- Download har2pcap.py, python3 har2pcap.py www.ikeriri.ne.jp.har www.ikeriri.ne.jp.pcap
- har2pcap accepts IPv4 address only, we modified it to accept IPv6 address. (GPT is great ☺

```
PS C:\Users\megumi\Desktop> python3 .\har2pcap.py .\www.ikeriri.ne.jp.har www.ikeriri.ne.jp.pcap
couldn't import pybase64 falling back to normal base64
src: 192.168.1.100 dst: 192.0.2.1 sport: 18195 dport: 443
```

Decrypt HTTPS without SSLKEYLOGFILE

- Har2pcap is not perfect, but we can decrypt to dissect HTTPS traffic without SSLKEYLOGFILE.
- We can utilize tons of Wireshark's rich functions to analyze har file (but lower-layer information is simulated and omitted such as TLS/TCP info



Chrome integration with Wireshark

please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip



Chrome Integration with Wireshark

- Chrome and modern browsers are the platform of your computing; it is a time for Integration.
- The simplest integration is a custom URL Scheme
- If you click mailto:megumi@ikeriri.ne.jp in your browser, a new message screen shows up in your mail application; it is a custom URL scheme.
- We can launch Wireshark in your Chrome using a custom URL scheme with simple steps

Creating reg file for Wireshark

- Microsoft provides handler for URL activation
  https://learn.microsoft.com/en-us/windows/apps/develop/launch/handle-uri-activation
- We create a reg file (wireshark.reg) in Windows11

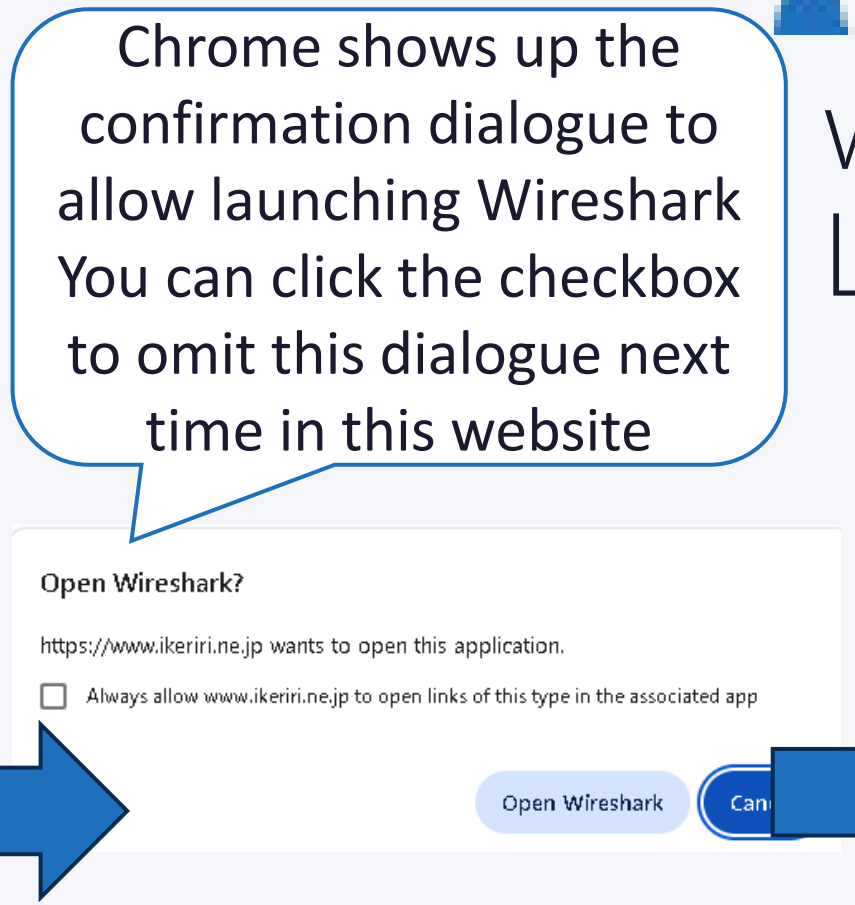
```
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT¥wireshark]
@="URL:Wireshark Protocol"
"URL Protocol"=""
[HKEY_CLASSES_ROOT¥wireshark¥shell]
[HKEY_CLASSES_ROOT¥wireshark¥shell¥open]
[HKEY_CLASSES_ROOT¥wireshark¥shell¥open¥command]
@="¥"C:¥¥Program Files¥¥Wireshark¥¥Wireshark.exe¥""
```

"URL Protocol" Indicates Custom Protocol Handler

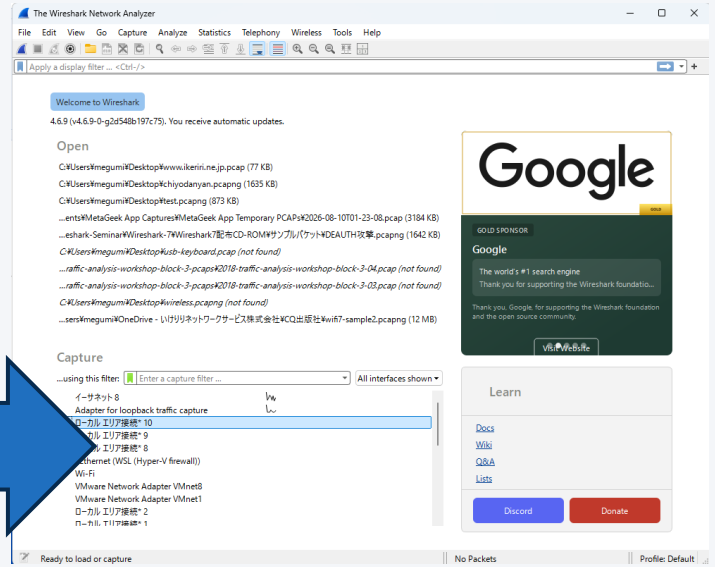
Register custom Protocol Handler

- Double-click wireshark.reg to register a custom protocol handler for Wireshark





- Let’s try wireshark:// Link in Chrome



Update reg file and open Wireshark with parameter

- We need a PowerShell launcher script to send parameters from the URL to Wireshark

```
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT¥wireshark]
@="URL:Wireshark Protocol"
"URL Protocol"=""

[HKEY_CLASSES_ROOT¥wireshark¥shell¥open¥command]
@="¥"C:¥¥Windows¥¥System32¥¥WindowsPowerShell¥¥v1.0¥¥powershell.exe
¥" -NoProfile -ExecutionPolicy Bypass -File
¥"C:¥¥Users¥¥megumi¥¥Desktop¥¥launcher.ps1¥" ¥"%1¥""
```

Update wireshark1.reg file
@= launch powershell and
filename is launcher.ps1 with %1
(includes http://wireshark/param)

Parameter-based launcher launcher.ps1

Wireshark –i interface –k –Y Display Filter

```
param(
    [string]$url
)
$Wireshark = "C:¥Program Files¥Wireshark¥Wireshark.exe"
$Uri = New-Object System.Uri($url)
$HostName = $Uri.Host
$Filter = “ip.host==$HostName"
Write-Host "URL      = [$url]"
Write-Host "HostName = [$HostName]"
Write-Host "Filter   = [$Filter]"
$Arguments = "-i Wi-Fi -k -Y `"$Filter`""
Start-Process -FilePath $Wireshark -ArgumentList $Arguments
```

Get URL wireshark://www.ikeriri.ne.jp/

Uri = www.ikeriri.ne.jp/

HostName = www.ikeriri.ne.jp

ip.host==www.ikeriri.ne.jp
from Uri.Host

-i Interface –k capture immediately
-Y Display Filter string

Register custom Protocol Handler

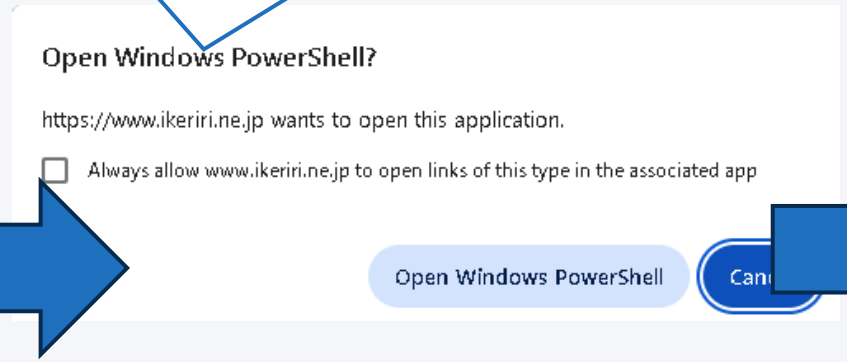
- Double-click wireshark1.reg to register a custom protocol handler for Wireshark
- Click the link, Wireshark start capturing with Display Filter

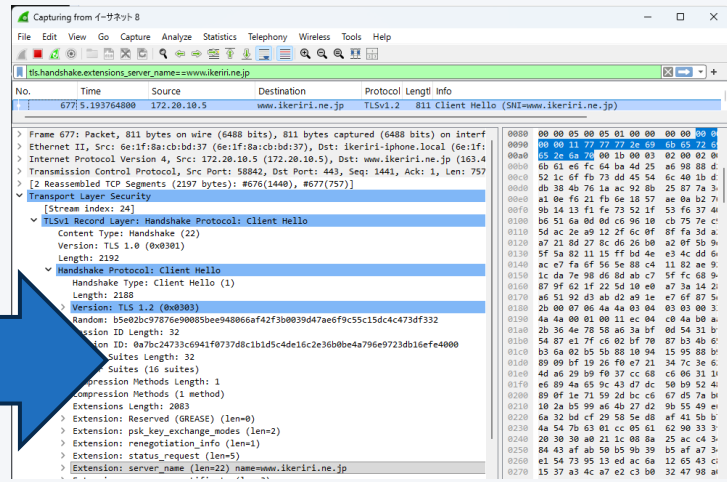


Open URL https://www.ikeriri.ne.jp/wireshark/en/cheer.html

Click Link “Open Wireshark with a href=wireshark://www.ikeriri.ne.jp/”

Chrome shows up the confirmation dialogue to allow launching Powershell lancher.ps1 gets URL wireshark://www.ikeriri.ne.jp as %1 and launch wireshark –i Wi-Fi –k –Y



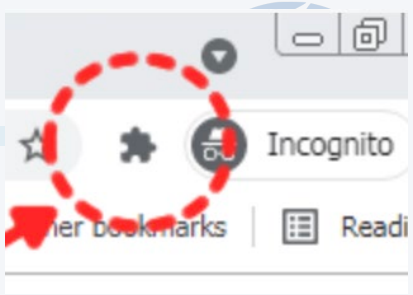


Chrome extension with Wireshark

please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip



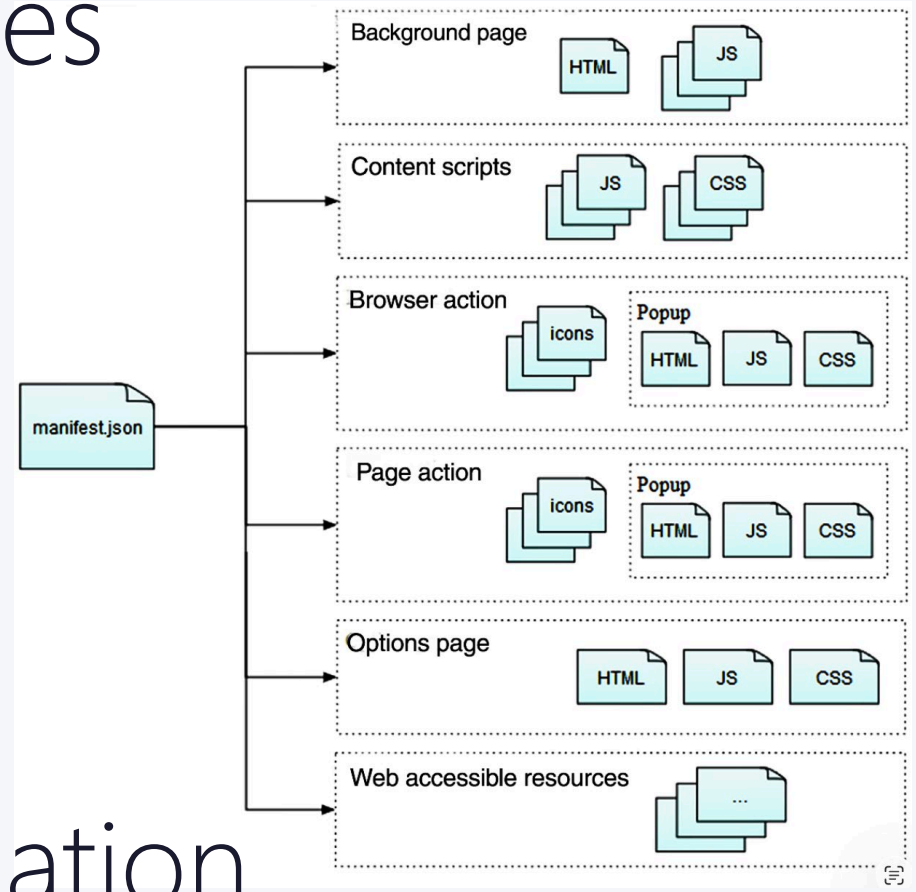
Chrome Extension (Manifest V3)



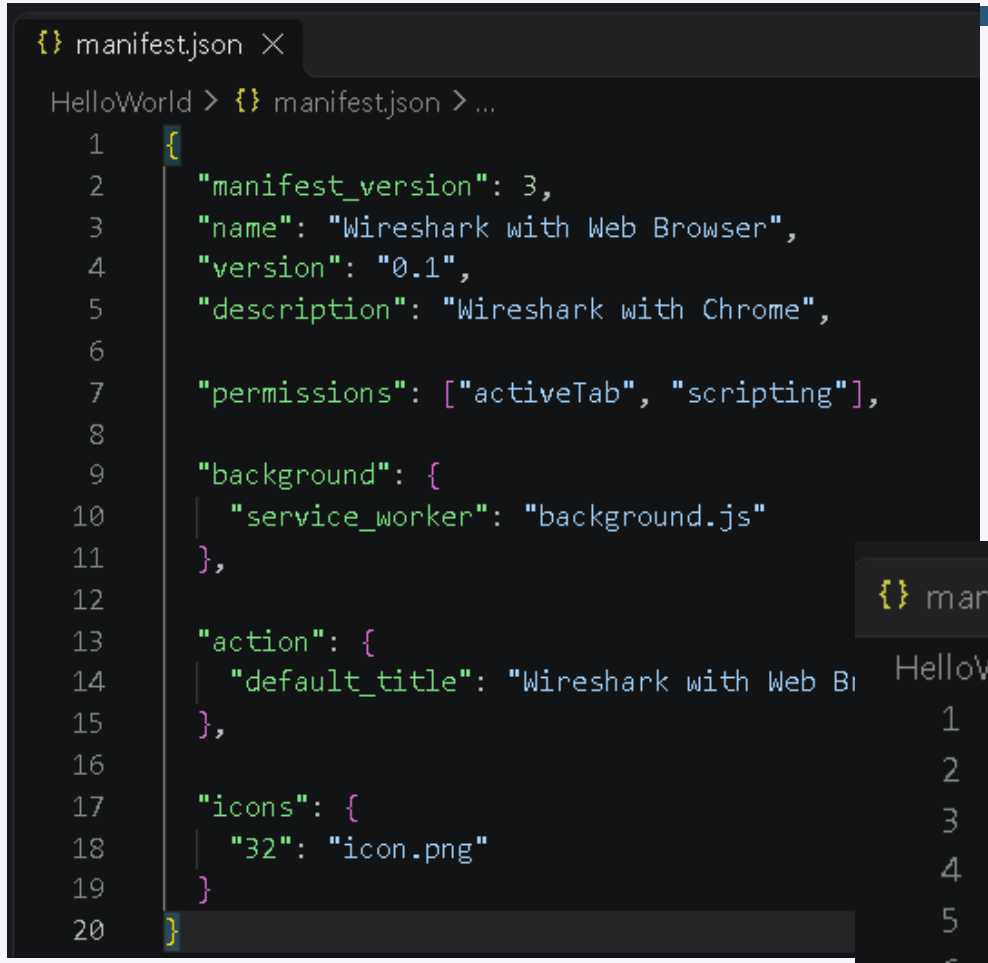
- Chrome Extension is a small program to add a function to Chrome, like an ad blocker
- The latest standard is Manifest V3 https://developer.chrome.com/docs/extensions/
- The 4MB size limitation if you post Chrome Web Store, but there is no limitation; it runs locally
- There are many useful APIs, we can call Native application, get information from webpage, etc.

Hello World (Manifest V3)

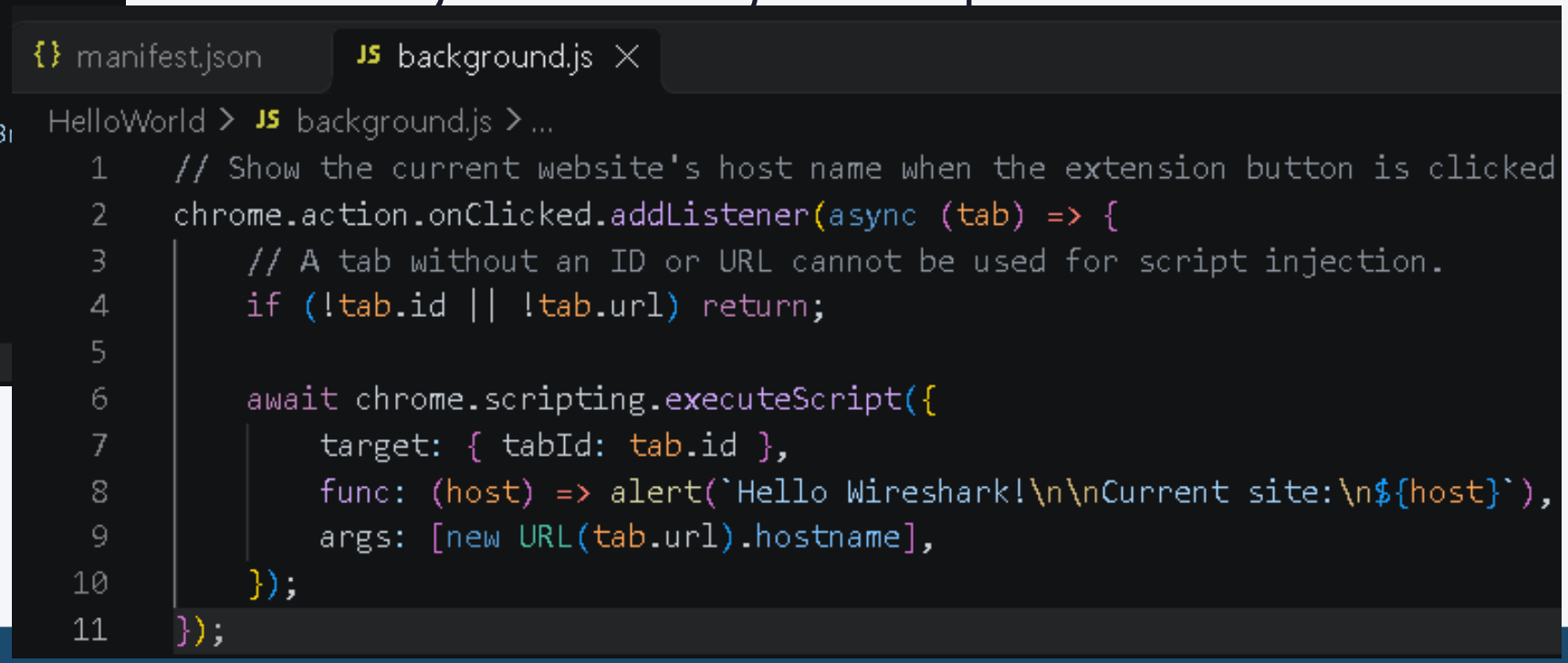
- We need to create manifest.json to declare the configuration of Chrome Extension, background.js for core processing and resources
- Create HelloWorld folder and open VisualStudio Code
- We test with the latest Chrome on Windows11 but you can use another OS and Web Browsers
- We use activeTab to get information



Manifest.json and background.js



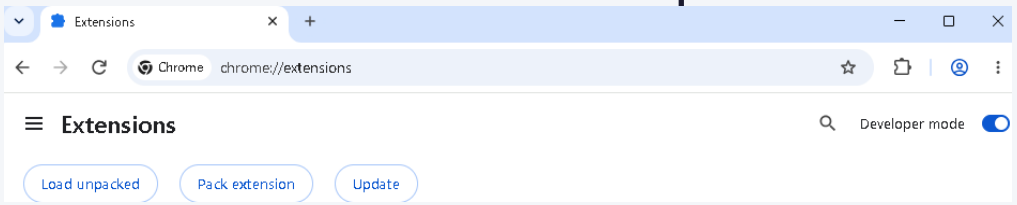
- Manifest.json is a specsheet of Chrome Extension
- We write javascripts in Background.js to process

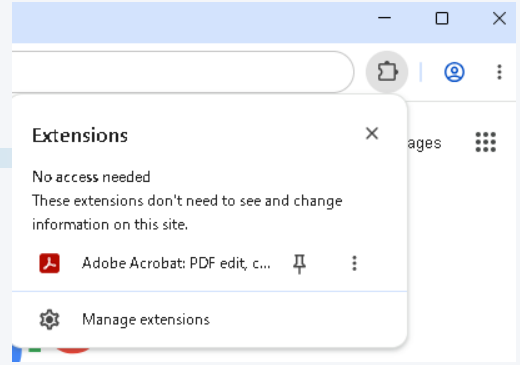


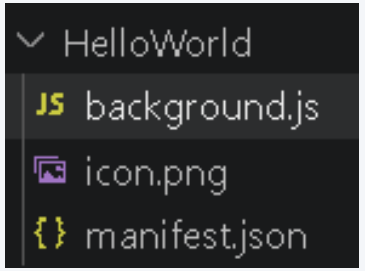
- activeTab.tab contains host

Hello World (Manifest V3)

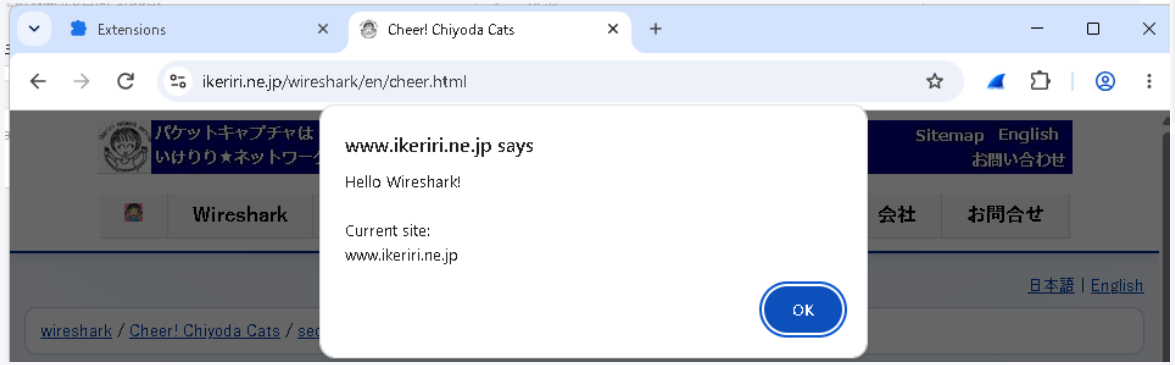
- Click Extensions -> Manage extensions -> turn on Developer mode switch

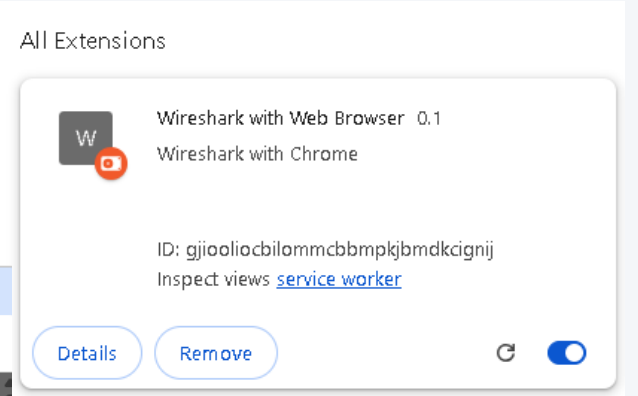






- Load unpacked to refer HelloWorld folder
- Extension added and icon appears
- Open https://www.ikeriri.ne.jp/wireshark/en/cheer.html
- Click the icon and message shows up





Hello Wireshark(custom URL scheme)

- Then, let's start integration with Wireshark
- We use the custom URL scheme (wireshark://) and protocol handler (launcher.ps1) to call Wireshark and capture Client Hello
- We use contextMenus for right click function
- Create HelloWireshark folder and put files
- Manifest.json is almost the same as before

HelloWireshark > {} manifest.json > ...

```
{
  "manifest_version": 3,
  "name": "Capture with Wireshark",
  "version": "0.2",
  "description": "Starts Wireshark for the current website",
  "permissions": [
    "contextMenus", "activeTab", "scripting"
  ],
  "background": {
    "service_worker": "background.js"
  },
  "icons": {
    "32": "icon.png"
  }
}
```

background.js of Hello Wireshark

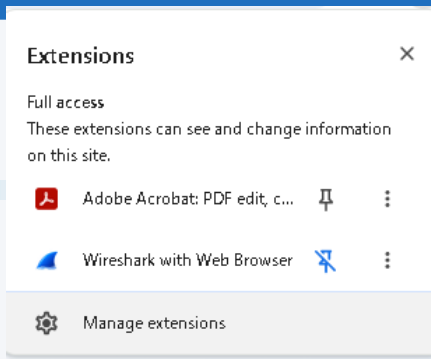
HelloWireshark > JS background.js > chrome.contextMenus.onClicked.addListener() callb

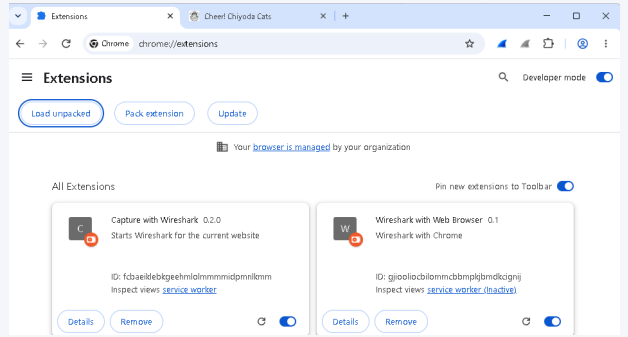
```
3   chrome.runtime.onInstalled.addListener(() => {
4     chrome.contextMenus.removeAll(() => {
5       chrome.contextMenus.create({
6         id: MENU_ID,
7         title: "Capture with Wireshark",
8         contexts: ["page"],
9         // Show this menu only on regular web pages.
10        documentUrlPatterns: ["http://*/*", "https://*/*"]});
11    });
12  });
13  chrome.contextMenus.onClicked.addListener(async (info, tab) => {
14    if (info.menuItemId !== MENU_ID || !tab?.id || !tab.url) {
15      return;}
16    try {
17      await chrome.scripting.executeScript({
18        target: { tabId: tab.id },
19        func: (host) => {
20          window.location.href = `wireshark://${host}/`;},
21        args: [new URL(tab.url).hostname]
22      });
23    } catch (error) {
24      console.error("Failed to start Wireshark:", error);}
25  });
```

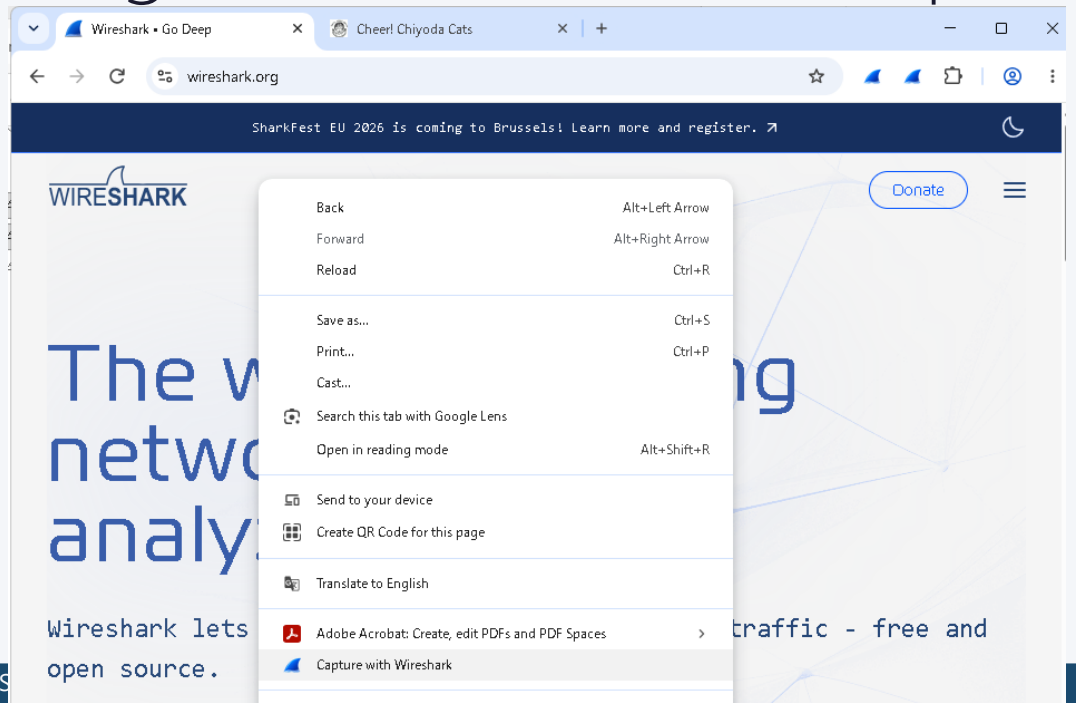
- chrome.runtime.onInstalled.addListener() for detecting right-click
- Use chrome.contextMenus.create() to create right-click menus for capturing the current page
- We write a process to select item as chrome.contextMenus.onCliked.addListener() with handling Exception
- When we open the webpage, right click and select „Capture with Wireshark“ in ContextMenu, wireshark://${host}/ is opened in your Chrome using custom URL scheme and protocol handler

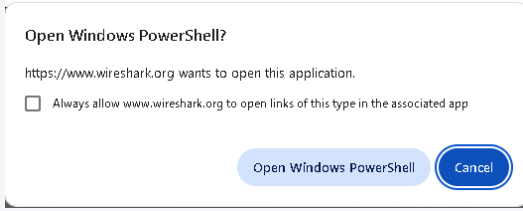
Hello Wireshark extension test

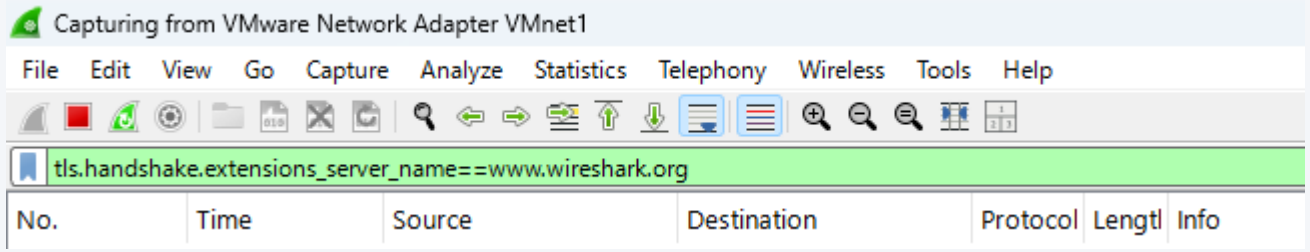
- Click Extensions -> Manage extensions
- Load unpacked and refer to the HelloWireshark folder
- Extension added and icon appears
- Open https://www.wireshark.org/ right-click and select Capture with Wireshark
- launcher.ps1 called and launch Wireshark with Display filter of hostname











Content Script to inject content into webpage

- We need to reload webpage if we want to get packets after Wireshark has been launched.
- How about adding “Capture” button into the webpage? ContentScript will inject content into the current webpage and add JavaScripts
- Create CaptureWireshark folder, set files as manifest.json, background.js, icon, content.js and content.css in this time

Manifest.json, background.js and content.css

CaptureWireshark > {} manifest.json > ...

```
{
    "manifest_version": 3,
    "name": "Wireshark with Web Browser",
    "version": "0.3",
    "description": "Capture with Wireshark",
    "background": {
      "service_worker": "background.js"
    },
    "content_scripts": [
      {
        "matches": ["http://*/*", "https://*/*"],
        "js": ["content.js"],
        "css": ["content.css"]
      }
    ],
    "icons": {
      "32": "icon.png"
    }
}
```

CaptureWireshark > # content.css > #wireshark-capture-button:ho

```
#wireshark-capture-button {
    position: fixed;
    right: 20px;
    bottom: 20px;
```

- We add “content_scripts” for the http/https webpage to inject contents.js scripts using content.css stylesheet
- content.css is the outlook for the hover capture button
- We do not need to write anything in background.js we just write content.js

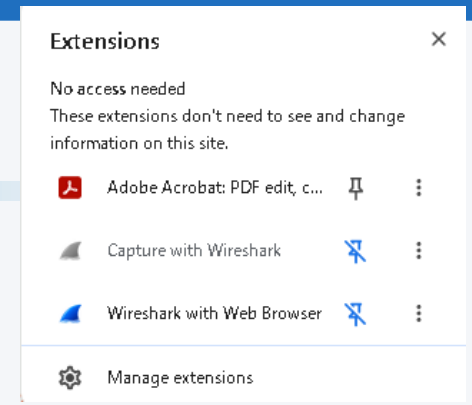
content.js (injection code for the current webpage)

CaptureWireshark > JS content.js > ...

```
1   // Create Capture button
2   const button = document.createElement("button");
3   button.id = "wireshark-capture-button";
4   button.textContent = "Capture with Wireshark";
5
6   button.addEventListener("click", () => {
7       // Get hostname of the current web page
8       const hostName = window.location.hostname;
9       // Create Custom URL
10      const wiresharkUrl =
11          "wireshark://" + hostName + "/";
12      // Launch Wireshark
13      window.location.href = wiresharkUrl;
14      // Reload the page to generate new packets
15      setTimeout(() => {window.location.reload();}, 1500);
16  });
17
18  document.body.appendChild(button);
```

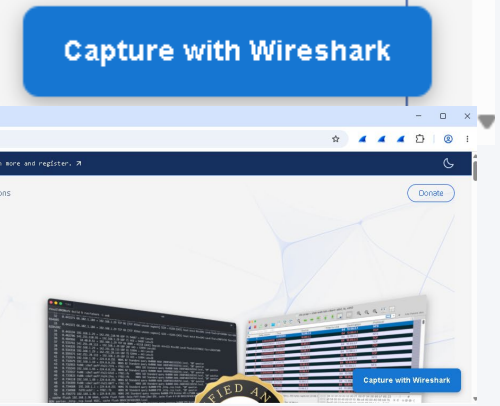
- document.createElement is used for creating button
- button.addEventListener() is used for event-driven;
- get hostname, create URL of “wireshark://hostname” and open the URL to launch Wireshark and reload the webpage itself

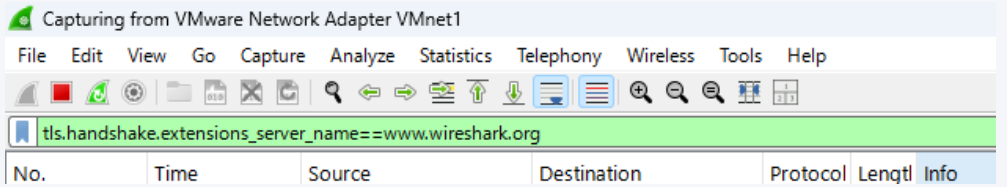
Hello Wireshark extension test





- Click Extensions -> Manage extensions
- Load unpacked to refer CaptureWireshark folder
- Extension added and icon appears
- Ctrl+Reload to reopen https://www.wireshark.org/ and scroll down
- We can find „Capture with Wireshark“ hover button!!
- Click „Capture with Wireshark“, URL „wireshark://www.wireshark.org“ called and launch launcher.ps1, Wireshark starts capturing with Display Filter!!
- It is useful in daily debugging!!





USE WIRESHARK

Thank you for your participation!
Please complete the survey



please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip

Appendix
Native Messaging API for integration between Wireshark and Web browser

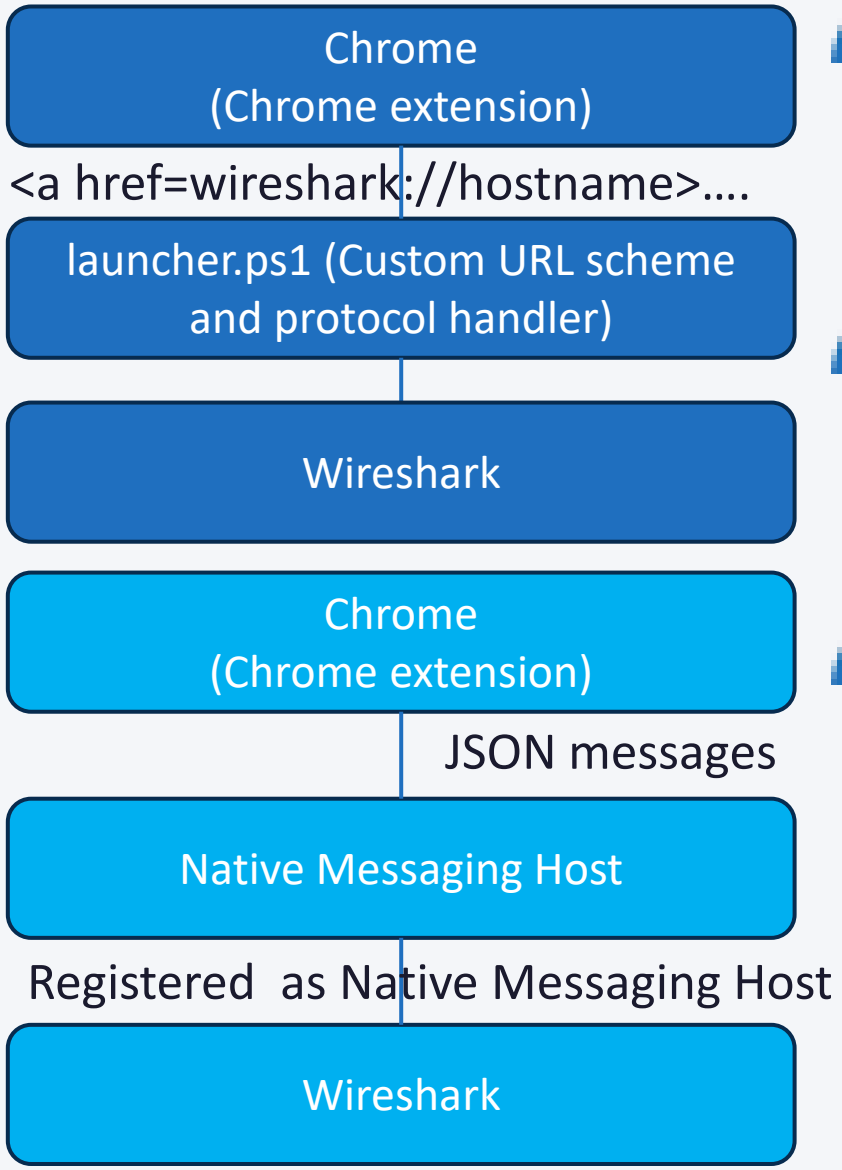
please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip



Appendix: Native Messaging API

- Writing small amount of code, we can utilize Chrome with Wireshark by TLS decryption, HAR file debugging, Chrome Integration such as the custom URL scheme, protocol handler and Chrome extensions.
- Native Messaging allows a Chrome Extension to communicate with a native application running on the local OS, Native Messaging API offers deep integration between Wireshark and Web Browser

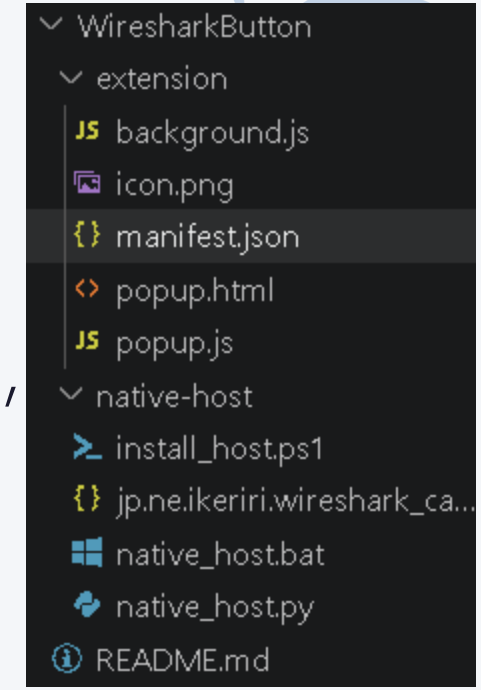
Appendix: Native Messaging API



- Last time, We used Chrome's custom URL scheme (wireshark://) and protocol handler (launcher.ps1) to call Wireshark
- Native Messaging allows a Chrome Extension to communicate with a native application running on the local OS.
- The native application is registered as a Native Messaging Host. Chrome starts the host as a separate process and communicates with the host through stdin and stdout

Appendix: WiresharkButton extension (1)

- Let's demonstrate the WiresharkButton extension, the sample of Native Message
- There are two folders; extension and native-host
- The extension folder is the Chrome extension as before, manifest.json is the specification of this extension, background.js is known as "service_worker" popup.html is a user interface of this extension popup.js gets the current website and creates a request
- native-host folder contains install_host.ps1 for registration of NativeHost at the installation; native_host.bat is the batch file called by the Chrome extension, and native_host.bat calls native_host.py to launch Wireshark. native_host.py communicates with the Chrome



Appendix: WiresharkButton extension (2)

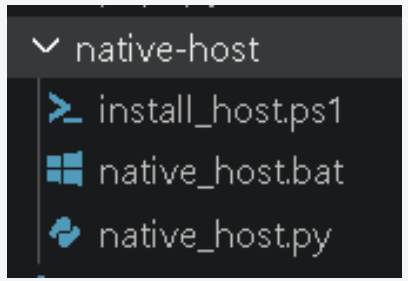
WiresharkButton > extension > {} manifest.json > ...

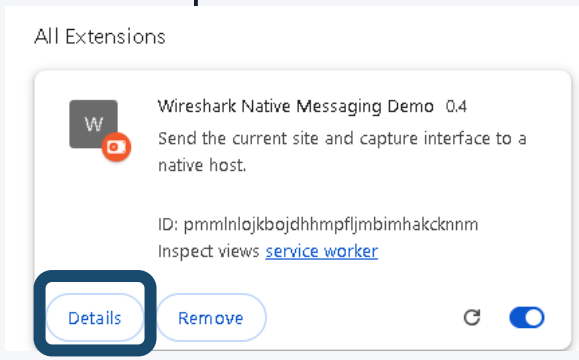
```
{
  "manifest_version": 3,
  "name": "Wireshark Native Messaging Demo",
  "version": "0.4",
  "description": "Send the current site and capture interface to a native host.",
  "permissions": ["activeTab", "tabs", "nativeMessaging"],
  "background": {"service_worker": "background.js"},
    "icons": {"32": "icon.png"},
  "action": {
    "default_title": "Wireshark Capture",
    "default_popup": "popup.html"}
}
```

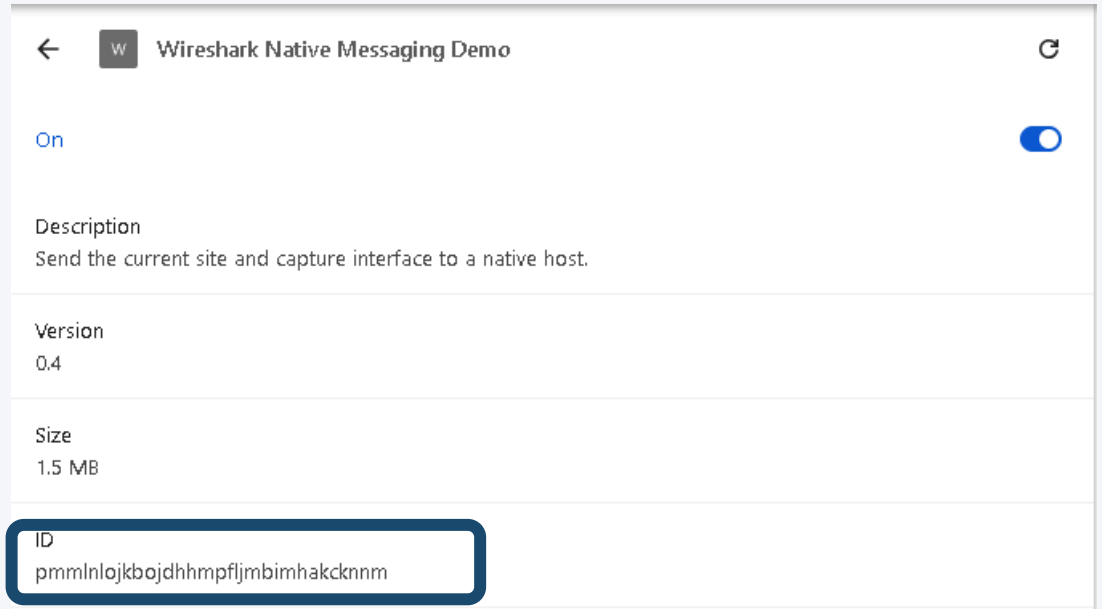
- Open WiresharkButton folder with VSCode, check manifest.json
- You can find "tabs", nativeMessaging in "permission" to use Natve Messaging
- We declare "service_worker" as "background.js" for background service in Chrome Manifest V3 specification
- We create a user interface of the extension this time, so we declare "action" for clicking the extension icon, we write popup.html in details

Appendix: WiresharkButton extension (3)

- We need to get the Chrome extension ID for registering Native Host
- Click Extensions -> Manage extensions -> Load unpacked to refer extension folder of WiresharkButton
- Click Details button of the Extension, copy ID (pmmlnlojkbojdhhmpfljmbimhakcknnm)
- We use this ID for registration of Native Host
- Open native-host folder we use install_host.ps1 before executing Chrome extension







Appendix: WiresharkButton extension (4)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\megumi> cd .\Desktop\WiresharkButton\
PS C:\Users\megumi\Desktop\WiresharkButton> py -3 --version
Python 3.14.3
PS C:\Users\megumi\Desktop\WiresharkButton> Set-ExecutionPolicy -Scope Process Bypass
```

```
.\native-host\install_host.ps1 -ExtensionId pmmlnlojkbojdhhmpfljmbimhakcknnm
```

- Open Powershell and move the WiresharkButton folder
- py -3 --version
  to check Python availability
- Set-ExecutionPolicy -Scope Process Bypass to allow scripts
- .¥native-host¥install_host.ps1 -ExtensionId <your Extension ID to register your extension

```
PS C:\Users\megumi\Desktop\WiresharkButton> .\native-host\install_host.ps1 -ExtensionId pmmlnlojkbojd

Registered: jp.ne.ikeriri.wireshark_capture
Manifest:   C:\Users\megumi\Desktop\WiresharkButton\native-host\jp.ne.ikeriri.wireshark_capture.json
PS C:\Users\megumi\Desktop\WiresharkButton>
```

- install_host.ps1 register the registry key, HKCU¥Software¥Google¥Chrome¥NativeMessagingHosts¥jp.ne.ikeriri.wireshark_capture

Appendix: WiresharkButton extension (5)



- Open registry editor to confirm HKEY_CurrentUSER¥Software¥Google¥Chrome¥NativeMessagingHosts¥jp.ne.ikeriri.wireshark_capture is registerd with your Desktop's json file

WiresharkButton > native-host > {} jp.ne.ikeriri.wireshark_capture.json > ...

```
{
  "name": "jp.ne.ikeriri.wireshark_capture",
  "description": "Wireshark Capture Native Messaging Host",
  "path": "C:\\Users\\megumi\\Desktop\\WiresharkButton\\native-host\\native_host.bat",
  "type": "stdio",
  "allowed_origins": [
    "chrome-extension://pmmlnlojkbojdhhmpfljmbimhakcknnm/"
  ]
}
```

- install_host.ps1 creates jp.ne.ikeriri.wireshark_capture.json file
- This json contains "type", "path" and "allowed_origins"

Appendix: WiresharkButton extension (6)

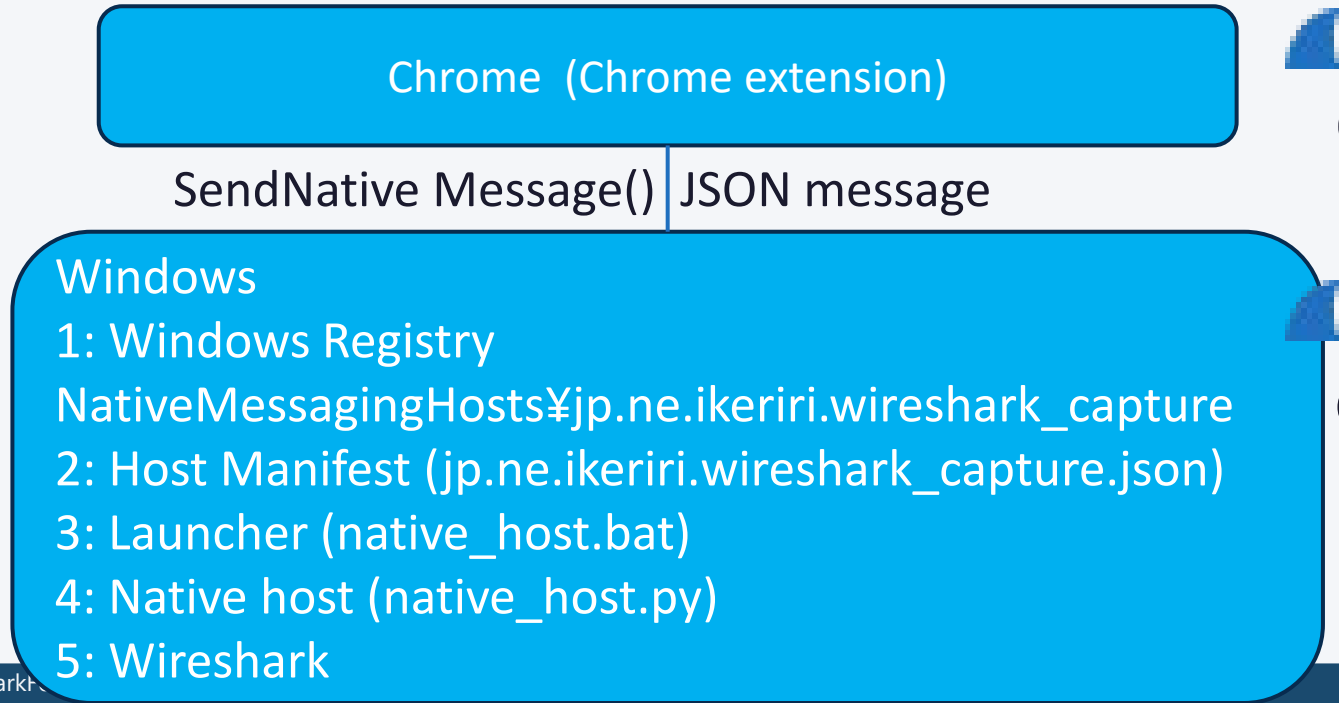
```
4    "path": "C:\\Users\\megumi\\Desktop\\WiresharkButton\\native-host\\native_host.bat",
```

- Native Host's path is C:¥Users¥megumi¥Desktop¥WiresharkButton¥native-host¥native_host.bat and this batch just call native_host.py

WiresharkButton > native-host > native_host.bat

```
1  @echo off
2  REM Launcher used by Chrome: runs the Python native host next to this file.
3  py -3 "%~dp0native_host.py"
```

- native_host.py calls Wireshark



- Flow diagram is here
  Chrome extension sends JSON message (SendNativeMessage)
- Windows refer registry and check the Host Manifest, call native_host.bat by "path" then native_hott.py call Wireshark

Appendix: WiresharkButton extension (7)

```python
"""Demo native messaging host: starts Wireshark filtered to the requested site."""
import json, re, struct, subprocess, sys

WIRESHARK = r"C:\Program Files\Wireshark\Wireshark.exe"

def read_message():
    # Chrome sends a 4-byte little-endian length followed by UTF-8 JSON.
    length, = struct.unpack("<I", sys.stdin.buffer.read(4))
    return json.loads(sys.stdin.buffer.read(length))

def send_message(message):
    data = json.dumps(message).encode("utf-8")
    sys.stdout.buffer.write(struct.pack("<I", len(data)) + data)
    sys.stdout.buffer.flush()

try:
    request = read_message()
    host, interface = request["host"], request.get("interface", "")
    if not re.fullmatch(r"[A-Za-z0-9.-]+", host):
        raise ValueError("Invalid host")

    # -i: interface, -k: start capturing now, -Y: filter on the TLS server name.
    args = [WIRESHARK] + (["-i", interface] if interface else [])
    subprocess.Popen(args + ["-k", "-Y", f"ip.host ={host}"])
    send_message({"status": "started", "host": host, "interface": interface or "default"})
except Exception as e:
    send_message({"status": "error", "message": str(e)})

```

- native_host.py read the JSON message from Chrome as standard input (read_message), get “interface” and “host” parsing the JSON, call Wireshark with the interface and Display Filter. (wireshark –i –Y)
- native_host.py also send JSON message to Chrome (send_message)

Appendix: WiresharkButton extension (8)

WiresharkButton > extension > popup.html > ...

```
2   <html lang="en"><head><meta charset="utf-8"><title>Wireshark Capture</title>
6   </style></head><body><main>
16  </div>
17  <script src="popup.js"></script></main></body></html>
```

WiresharkButton > extension > popup.js > ...

```
1   // Popup: shows the current site, asks the background worker to start a capture, shows the reply.
2   const $ = (id) => document.getElementById(id);
3   let tabId;
4
5   chrome.tabs.query({active: true, currentWindow: true}).then(([tab]) => {
6     tabId = tab.id;
7     try {
8       $("host").value = new URL(tab.url).hostname;
9     } catch {
10      $("start").disabled = true;
11      $("status").textContent = "Open an HTTP or HTTPS page first.";
12    }
13  });
14
15  $("start").onclick = () => {
16    const request = {command: "capture", host: $("host").value, interface: $("iface").value.trim()};
17    $("outgoing").textContent = JSON.stringify(request, null, 2);
18    $("incoming").textContent = "Waiting for native host...";
19    $("status").textContent = "Sending...";
20    $("start").disabled = true;
21
22    chrome.runtime.sendMessage({type: "START_CAPTURE", tabId, ...request}, (response) => {
23      response ??= {status: "error", message: chrome.runtime.lastError?.message};
24      $("incoming").textContent = JSON.stringify(response, null, 2);
25      $("status").textContent = response.status === "started" ? "Capture started" : "Capture failed";
26      $("start").disabled = false;
27    });
28  };
```

- popup.html is the main user interface of this extension. This HTML shows up when we click the extension icon
- popup.js is the main contents of user interface.
- The popup gets the current website, creates a capture request, sends it to the background service worker, and displays the response.

Appendix: WiresharkButton extension (9)

WiresharkButton > extension > JS background.js > ...

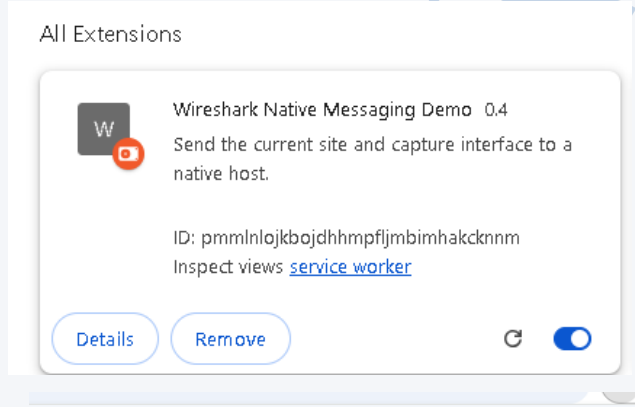
```
1   // Background service worker: relays capture requests from the popup to the
2   // native messaging host (which launches Wireshark).
3   // Must match the host name registered by install_host.ps1.
4   const NATIVE_HOST = "jp.ne.ikeriri.wireshark_capture";
5   chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
6     if (message?.type !== "START_CAPTURE") return;
7     const request = {
8       command: "capture",
9       host: message.host,
10      interface: message.interface || ""
11    };
12    chrome.runtime.sendNativeMessage(NATIVE_HOST, request, (response) => {
13      if (chrome.runtime.lastError) {
14        sendResponse({status: "error", message: chrome.runtime.lastError.message});
15        return;
16      }
17      if (response?.status === "started" && message.tabId) {
18        // Reload the page shortly after Wireshark starts so the capture sees fresh traffic.
19        setTimeout(() => chrome.tabs.reload(message.tabId), 1200);
20      }
21      sendResponse(response || {status: "error", message: "No response"});
22    });
23    // Keep the message channel open for the asynchronous sendResponse.
24    return true;
25  });
```

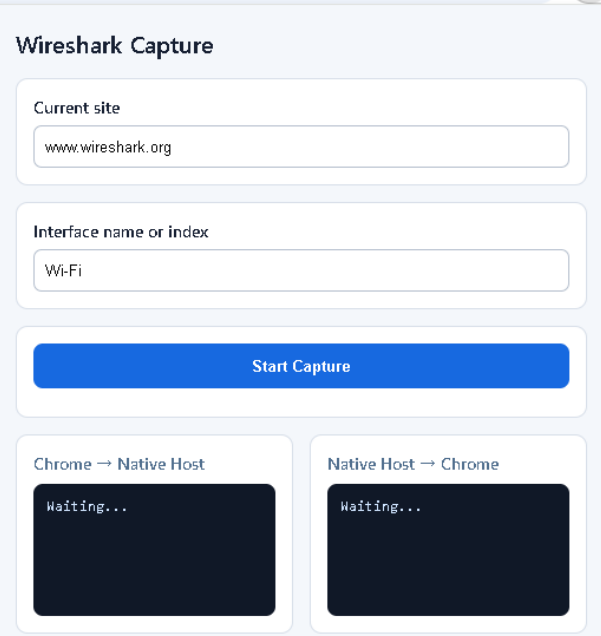
- background.js is called as “service_worker”; we write the service for this extension.

- background.js works as a bridge between the popup and the Native Messaging Host. It receives a capture request from popup.js as “START_CAPTURE”, sends the request JSON to the Native Messaging Host as sendNativeMessage, receives the response as “started” from NativeHost, and returns it to the popup.

Appendix: WiresharkButton extension (10)

- That's it, the codes are bigger than before
- Let's try, click Extensions -> Manage extensions Reload Wireshark Native Messaging Demo
- Open the website https://www.wireshark.org/ and Click extension button, popup.html shows up, check the current site is www.wireshark.org, You can change Interface name or index field (default value: Wi-Fi), then click Start Capture





Chrome → Native Host

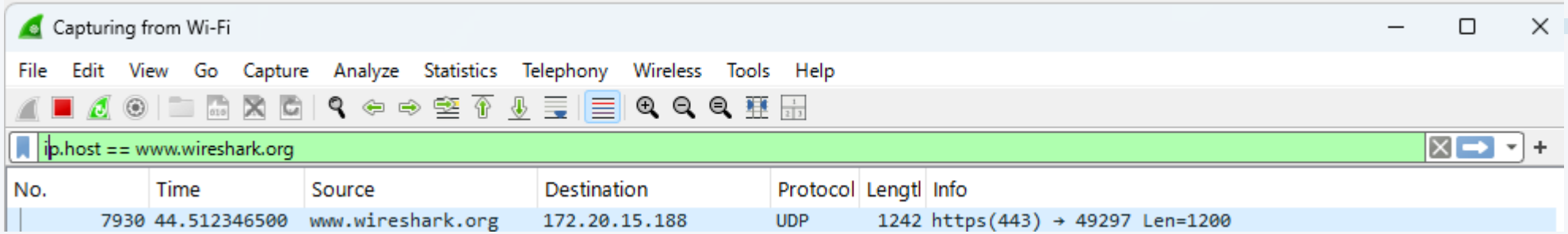
```
{
  "command": "capture",
  "host":
"www.wireshark.org",
  "interface": "Wi-Fi"
}
```

Native Host → Chrome

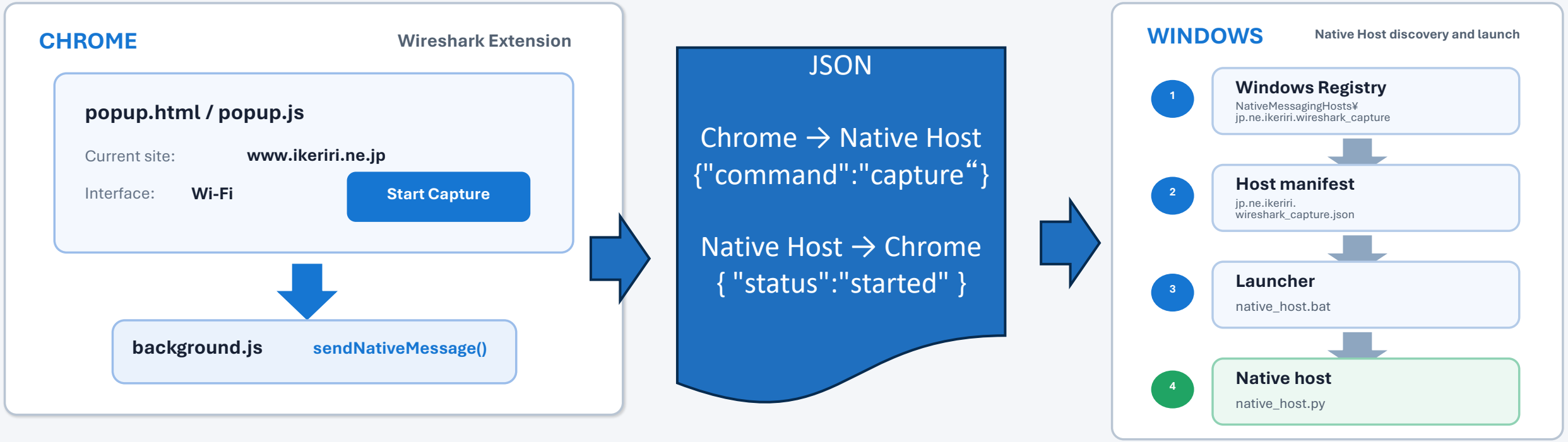
```
{
  "status": "started",
  "host":
"www.wireshark.org",
  "interface": "Wi-Fi"
}
```

- Chrome sends JSON to Native Host and Native Host send back JSON to Chrome

Appendix: WiresharkButton extension (11)



- Wireshark starts capturing at the interface you set with Display Filter!!



- Turn your browser into a front end for Wireshark/tshark !!

USE WIRESHARK

Thank you for your participation!
Please complete the survey



please download the trace files and scripts at
https://www.ikeriri.ne.jp/sharkfest/ikeriri26eu.zip