CyberSecurity CW2 Individual Analysis

Xiao Jun Huang

April 8, 2022

Contents

1 Question1

With which websites does your browser communicate during the transaction? Are there any that worry you, or whose purpose/function you do not understand?

Since I installed ’AdBlock’ on my browser, there may not be that many activities being logged. Inside the whole interaction process, there are 69 entries that could be seen from the HAR file.
Along with the timeline (starting at 2022-03-23T12:22:12.975Z, ending at 2022-03-23T12:22:12.975Z), which not just involved the transaction process only, I went through everyone and pick up the important steps during the transaction, and analyzed each of them. Then basically, I got the flow of the whole process as described in (Figure 1). Detailed analysis are listed below. And in the following part, I would call these networking functions ’entry’, using ’...’ to simplify the parameters which are too long.
There are 5 IP addresses the browser communicated during the whole process:

- 52.224.31.34 (https://h.clarity.ms)
- 18.170.126.184 (https://p.yotpo.com)
- 104.22.62.160 (https://store.liverpoolfc.com)
- 104.18.250.34 (https://flex.cybersource.com)
- 198.217.251.250 (https://xxxxx.cardinalcommerce.com)
- 104.16.18.94 (https://cdnjs.cloudflare.com)

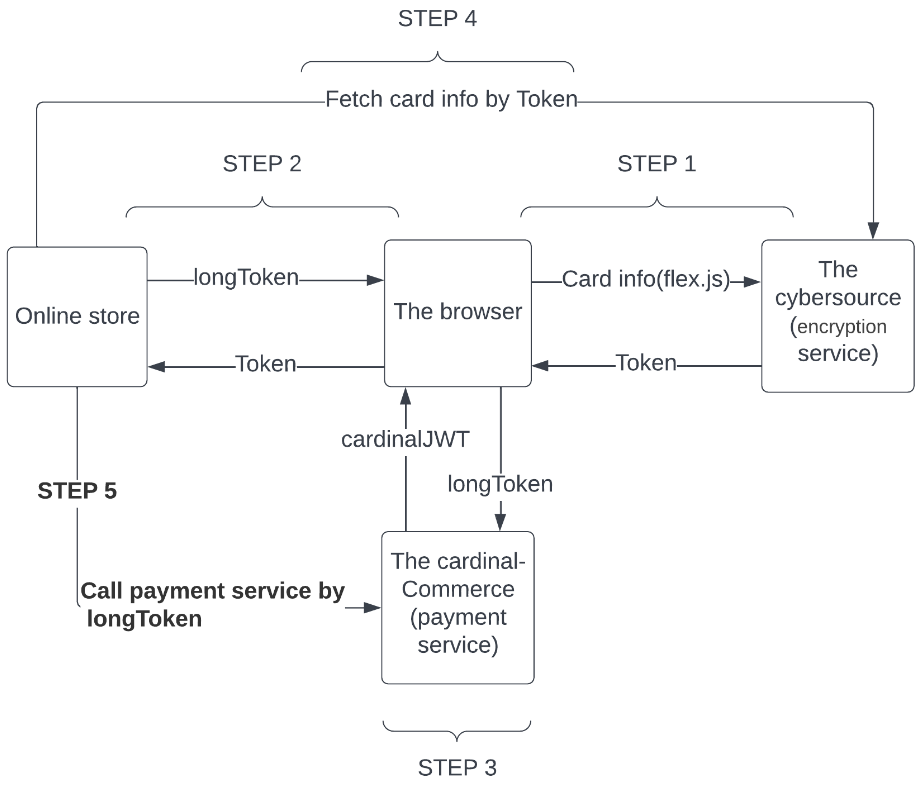


Figure 1: The Flow

1.1 52.224.31.34 (https://h.clarity.ms)

Among all the entries, this website occurs 13 times. And in every entry’s request header, there are two parameters ’origin=https://store.liverpoolfc.com’ and ’referer=https://store.liverpoolfc.com/’. Therefore, it reveals that inside the online store website, it integrated the service clarity(MS, 2022), which aims to capture how real people use the site. This could help them analyze people’s actions on the website and change their strategy for selling.

1.2 18.170.126.184 (https://p.yotpo.com)

This website occurs 29 times, and in all of these entries, the priority is 'low', the 'resourceType' is 'image' and the 'request method' is 'GET'. By checking the request and the response, it is clear that these 29 entries are requesting image or gif resources from the website. So this website provides cloud service for media resource online storage(yopto, 2022), and some other services.

1.3 104.22.62.160 (https://store.liverpoolfc.com)

1.3.1 Entry No.6

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/rest/default/V1/carts/mine/set-payment-information
- Request Method : POST
- PostData : {"cartId":"4484648","paymentMethod":{"method":"chcybersource"}}
- Response : Status = 200, text = true

Inside this entry, there are 18 callFrames including a function named 'selectPaymentMethod', so basically, this entry tells the server which kind of payment type the user has chosen, which would bond with the cart.

1.3.2 Entry No.7

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/captcha/refresh/
- Request Method : POST
- PostData : {"formId":"payment_processing_request"}
- Response : Status = 200, text = {"imgSrc":"https://store.liverpoolfc.com/...1484d24eb3.png"}

This entry just helps refresh web page.

1.3.3 Entry No.8

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/rest/default/V1/carts/mine/totals?_=1648037983125
- Request Method : GET
- PostData : None
- Key in headers : refer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"grand_total":29.5,"base_grand_total":29.5,...}

This entry is also a subsequence of entry NO.6, it gets the information about price and others have to be shown on the web page from the server, the key should be the ID of this session or the ID of the cart, but I do not know where the '1648037983125' comes from, it seems like timestamp, but it cannot match with time this request been sent. This entry ensures that all the important information is from the back-end, which is calculated correctly. Since if the figure of price is calculated by the local js file, it would easily be changed by attackers, what if the buyer should pay just £10 but the figure showing on the price label is £1000, these mistakes could make the online store losing more and more customers.

1.3.4 Entry No.9

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/customer/section/load/?sections=messages%2Ccompany%2Cpersonal-data&force_new_section_timestamp=true&_=1648037983126
- Request Method : GET
- PostData : None
- Response : Status = 200, text = {"messages":{"messages":[],..."data_id":1648038162}

Mainly reload or check the section for purchasing.

1.3.5 Entry No.44

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/cybersource3ds/cca/requestToken
- Request Method : POST
- PostData : text = method=chcybersource...form_key=EQLwGIWOn6Wz1hum
- Key in headers : referer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"success":true,"token":"eyJ0e...2dA"}

This entry is requesting a token from the online store, see the 'STEP 2' in (Figure 1). It gives the 'Token' which requested from cybersource(cybersource, 2022a) in 'STEP 1' to the online store and gets the 'longToken' that is essential for 'STEP 3'. With 'ybersource3ds/cca/' in this URL and js files containing 'Magento' in the folder, I think this site used 'CardinalCommerce 3-D Secure' (Adobe, 2021) to protect your web store from fraud, reduce false declines, reduce manual review of orders, and improve authorizations.

1.3.6 Entry No.45

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/customer/section/load/?sections=messages%2Ccompany%2Cpersonal-data&force_new_section_timestamp=true&_=1648037983127
- Request Method : GET
- PostData : None
- Key in headers : referer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"messages":{"messages":[],..."data_id":1648038403}

This entry does the same thing as entry NO.9 does.

1.3.7 Entry No.61

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/rest/default/V1/carts/mine/payment-information
- Request Method : POST
- PostData : text = {"cartId":"4484648","billingAddress":..."token":"eyJra...PQ","sessionId":"1_3db0537b-d054-4c5d-bae8-2f95bca58979"}
- Key in headers : referer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"message":"Transaction has been declined. Please try again later."}

This entry is exactly calling the 'purchase' function, and in the parameters, there is information about the order. The 'cartId' and 'sessionId' are the key parameters for the online store to search the order inside their database. While I can not see the 'cvv number' and the 'card number', because the card info is already encrypted and sent to CyberSource (cybersource, 2022a) by js file flex (Microform, 2018). In return, CyberSource gives a 'token' back for following transaction, which is mathematically irreversible and replaces the card info. And the online store would get 'cvv' and 'card number' of mine by sending the 'token' to CyberSource. Once the online store gets the card info, it would communicate with cardinalCommerce(Cardinal, 2022) to do the real payment. And since our information is fake, the response is 'Transaction has been declined. Please try again later.', which is quite cautious, and did not cause information leak(CWE-200, 2021).

1.3.8 Entry No.62

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/captcha/refresh/
- Request Method : POST
- PostData : text = {"formId":"payment_processing_request"}
- Key in headers : referer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"imgSrc":"https:store.liverpoolfc.com...fe8c6390d3e95e4b.png"}

1.3.9 Entry No.63

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/rest/default/V1/carts/mine/totals?_=1648037983128
- Request Method : GET
- PostData : None
- Key in headers : referer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"grand_total":29.5,"base_grand_total":29.5,...}

1.3.10 Entry No.64

- SeverIPAddress : 104.22.62.160
- Priority : High
- URL : https://store.liverpoolfc.com/customer/section/load/?sections=cart%2Clast-ordered-items%2Cinstant-purchase%2Cammessages%2Cmessages%2Ccompany%2Cpersonal-data&force_new_section_timestamp=true&_=1648-037983129
- Request Method : GET
- PostData :
- Key in headers : referer = https://store.liverpoolfc.com/checkout/
- Response : Status = 200, text = {"cart":{"summary_count":1,"subtotalAmount":25,..."data_id":1648038410}}

Entry NO.62, NO.63 and NO.64 could be considered as page refreshing job after one payment failure.

1.4 104.18.250.34 (https://flex.cybersource.com)

1.4.1 Entry No.43

- SeverIPAddress : 104.18.250.34
- Priority : High
- URL : https://flex.cybersource.com/flex/v2/tokens
- Request Method : POST
- PostData : text = eyJraWQiOiIwM0VySGlPYk1...VQIiwiZW5jIjoiQTI1NkdDTSJ9
- Response : Status = 201, text = eyJraWQiOiIwM0VySGlPYk15NU...Oqmo0OJcRldSYT9Oi1PGTHPQ

This site only occurs once, the js file flex(Microform, 2018) helps encrypt the card info, and sends it to cyber-source(cybersource, 2022a), getting a 'token' back for the following transaction.

1.5 198.217.251.250 (https://xxxxx.cardinalcommerce.com)

1.5.1 Entry No.46

- SeverIPAddress : 198.217.251.250
- Priority : Low
- URL : https://songbird.cardinalcommerce.com/edge/v1/59...33d4189.songbird.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = too much to write down

Adding songbird(Simone, 2020) to the front end.

1.5.2 Entry No.47

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://centinelapi.cardinalcommerce.com/V1/Order/JWT/Init
- Request Method : POST
- PostData : {"BrowserPayload":{"Order":{"OrderDetails":{},...}
- Response : Status = 200, text = {"CardinalJWT":"eyJhb...uC_Y"}

1.5.3 Entry No.48

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://centinelapi.cardinalcommerce.com/V1/Order/JWT/Init
- Request Method : OPTIONS
- PostData : None
- Response : Status = 200

JWT creation(Simone, 2020).

1.5.4 Entry No.49

- SeverIPAddress : 198.217.251.250
- Priority : Low
- URL : https://songbird.cardinalcommerce.com/edge/v1/597...c33d4189.songbird.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = too much to write down

Adding songbird(Simone, 2020) to the front end again? The js url is different with entry NO.46.

1.5.5 Entry No.50

- SeverIPAddress : 198.217.251.250
- Priority : Veryhigh
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/V2/Br...lm&geolocation=false&origin=Songbird
- Request Method : POST
- PostData : text : nonce=bc484737-6805-489c-9479-e8aac4fb8f89
- Response : Status = 200

This entry tells 'cardinalcommerce'(Cardinal, 2022) it is going to purchase.

1.5.6 Entry No.52

- SeverIPAddress : 198.217.251.250
- Priority : Low
- URL : https://songbird.cardinalcommerce.com/edge/v1/597f4104d311c33d4189/3.597f4104d311c33d4189.songbird.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = too much to write down

Adding songbird(Simone, 2020) to the front end again? The js url is different with the former two.

1.5.7 Entry No.55

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/include...b0fc274afbf37412e.min.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = /* * Fingerprintjs2 1.5.1 - Modern & flexible browser fingerprint library...

1.5.8 Entry No.56

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/includes/js/profiler.min.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = window.Logger=function(){var...r={domain:"merchant",environment:"SBOX...

1.5.9 Entry No.57

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/includes/js/acsprofiler.min.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = window.acsprofiler=function(a){var d,i=1e4,s,l={start:function...

Seems like they are a kind of event ('fingerPrint') listener JS or preparation process for the checking out.

1.5.10 Entry No.58

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/V2/Browser/SaveBrowserData
- Request Method : POST
- PostData : text = {"BinConfigIdentifiers":null,"Cookies"..."BinSessionId":"bc484737-6805-489c-9479-e8aac4fb8f89"}
- Response : Status = 200, text = None

This entry tells the cardinalcommerce(Cardinal, 2022) the status including information of the browser at present.

1.5.11 Entry No.59

- SeverIPAddress : 198.217.251.250
- Priority : High
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/V2/Bin/Enabled
- Request Method : POST
- PostData : text = {"bin":"526941","orgUnitId":"5f1654934253d447d610136a"}
- Response : Status = 200, text = {"ConfigsEnabled":false,"MethodUrlEnabled":false}

1.5.12 Entry No.60

- SeverIPAddress : 198.217.251.250
- Priority : OPTIONS
- URL : https://geo.cardinalcommerce.com/DeviceFingerprintWeb/V2/Bin/Enabled
- Request Method : POST
- PostData : text = {"bin":"526941","orgUnitId":"5f1654934253d447d610136a"}
- Response : Status = 200

Entry NO.59 and NO.60 are trying to enable some function, and NO.59 is the first try, No.60 backs up NO.59.

2 104.16.18.94 (https://cdnjs.cloudflare.com)

2.1 Entry No.53

- SeverIPAddress : 104.16.18.94
- Priority : High
- URL : https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = /*! jQuery v3.5.1 — (c) JS Foundation and other ...

CDN is a network service provider(Cloudflare, 2022).

2.2 Entry No.54

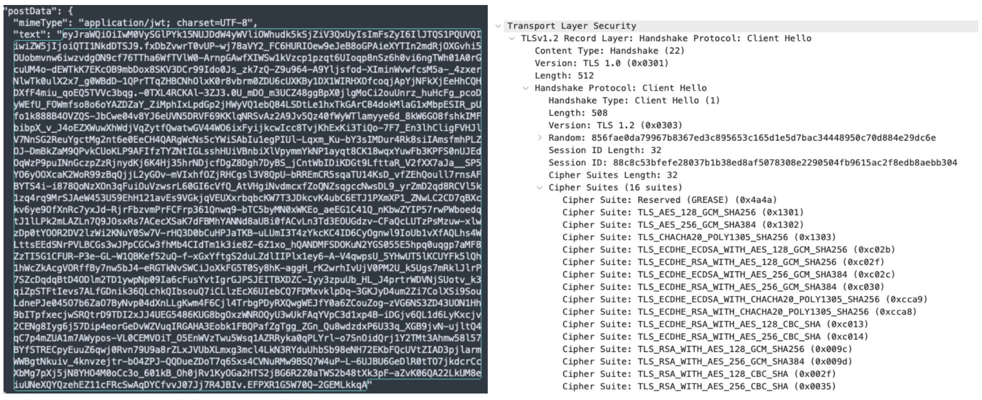
- SeverIPAddress : 104.16.18.94
- Priority : High
- URL : https://cdnjs.cloudflare.com/ajax/libs/json3/3.3.2/json3.min.js
- Request Method : GET
- PostData : None
- Response : Status = 200, text = /*! JSON v3.3.2 — http://bestiejs.github.io/json3 — Copyright 2012-2014, Kit Cambridge — http...

CDN is a network service provider(Cloudflare, 2022).

3 Question2

Looking at the logs, to which sites does your payment card number get sent, and how is it protected in transit? You should quote the relevant part of the logs, but should replace any plain-text card number and any other identifying/sensitive data, e.g. by NNNN NNNN NNNN NNNN, before quoting.

First of all, there is no plain-text form 'cardNumber' inside the HAR file. Based on the analysis of question 1, entry NO.61 did the 'purchasing' thing. While inside the 'postData' of this entry, I can not see the 'cvv' and the 'card number', since 'Token' that is returned in 'STEP 1' contains the card info, see the (Figure1).
So the payment card number is sent to the site cybersource (cybersource, 2022a) '104.18.250.34 (https://flex.cybersource.com)'. And the card info is collected by flex.js (Microform, 2018), which replaced the sensitive card number input field with a secure iframe (hosted by CyberSource). And in transit, it is well protected by TLSv1.2 protocol(TLS, 2022), which could be found in the Wireshark file. In return, the browser gets the 'Token'(cybersource, 2022b). As for the logs, it is encrypted as (Figure 2a) when the browser is call the URL 'https://flex.cybersource.com/flex/v2/tokens'.



```
"postData": {
  "mimeType": "application/jwt; charset=UTF-8",
  "text": "eyJraWQiOiIwM0VySGlPYk15NUJDdW4yWVliOWhudk5kSjZiV3QxUyIsImFsZyI6IlJTQS1PQUVQI
  iwiZW5jIjoiQTI1NkdDTSJ9.fxDbZvwrT0vUP-wj78aVY2_FC6HURIOew9eJeB8oGPAieXYTIn2mdRjOXGvhi5
  OUobmvnw6iwzvdgON9cf76TTha6WfTVlW0-ArnpGAwfXIWSw1kVzcp1pzqt6UIoqp8nSz6h0vi6ngTWh01A0rG
  cuUM4o-dEWTkK7EKcOB9mbDox8SKV3DCr99Ido0Js_zk7zQ-Z9u964-A9Yljsfod-XIminWvwfcsM5a-_4zxer
  NlwTk0ulX2x7_g0WBdD-1QPrTTqZHBCNhOlxK0r8vbrm0ZDU6cUXKBy1DX1WIRHXOfcoqjApYjNFkXjEeHhCQH
  DXfF4miu_qoEQ5TVVc3bqg.-0TXL4RCKAl-3ZJ3.0U_mDO_m3UCZ48ggBpX0jlgMoCi2ouUnrz_huHcFg_pcoD
  yWEfU_FOWmfso8o6oYAZDZaY_ZiMphIxLpdGp2jHWyVQ1ebQ84LSDtLe1hxTkGArC84dokMlaG1xMbpESIR_pU
  fo1k888B4OVZQS-JbCwe04v8YJ6eUVN5DRVF69KKlqNRSvAz2A9Jv5Qz40fWyWTlamyye6d_8kW6GO8fshkIMF
  bibpX_v_J4oEZXWuwXhWdjVqZytfQwatwGV44WO6ixFyijkcwIcc8TvjKhExKi3TiQo-7F7_En3lhCligFVHJl
  V7NnSG2ReuYgctMg2nt6e0EeCH4QARgWcNs5cYWiSAbIu1egPIUl-Lqxm_Ku-bY3sIMDur4Rk8siIAmsfmhPLZ
  OJ-DmBkZaM9QPvkCUoKLP9AFIfzTYZNtIGLsshHUiVBnbiXlVpymmYkNP1ayqt8CK18wqxYuwFb3KPFS0nUJEd
  OqWzP9puINnGczpZzRjnydKj6K4Hj35hrNDjcfDgZ8Dgh7DyBS_jCntWbIDiKDGt9LfttaR_V2fXX7aJa__SP5
  YO6yOOXcaK2WoR99zBqQjjL2yGOv-mVIxhfOZjRHCgsl3V8QpU-bRREmCR5sqaTU14KsD_vfZEhQoull7rnsAF
  BYTS4i-i878QoNzXOn3qFuiOuVzwsrL60GI6cVfQ_AtVHgiNvdmcxfZoQNZsqgccNwsDL9_yrZmD2qd8RCVl5k
  1zq4rq9MrSJAeW453U59EhH121avEs9VGkjqVEUXxrbqbcKW7T3JDkcvK4ubC6ETJ1PXmXP1_ZNwLC2CD7qBXc
  kv6ye9OfXnRc7yxJd-RjrFbzvmPrFCFrp361Qnwq9-bTC5byMN0xWKEo_aeEG1C41Q_nKbwZYIP57rwPWboedq
  tJ1lLPk2mLAZLn7Q9JOsxRs7ACecXSaK7dFBMhYANNd8aUBi0fACvLn3Td3EOUGdzv-CFaQcLUTzPsMzuw-xlw
  zDp0tYOOR2DV2lzWi2KNuY0Sw7V-rHQ3D0bCuHPJaTKB-uLUmI3T4zYkcKC4ID6CyOgnwl9IoUb1vXfAQLhs4W
  LttsEEdSNrPVLBCGs3wJPpCGCw3fhMb4CIdTm1k3ie8Z-6Z1xo_hQANDMFSDOKuN2YGS055E5hpq0uqgp7aMF8
  ZzTI5G1CFUR-P3e-GL-W1QBKef52uQ-f-xGxYftgS2duLZdlIIPlx1ey6-A-V4qwpsU_5YHwUT5lKCUYFk5lQh
  1hWcZkAcgVORffBy7nw5bJ4-eRGTkNvSWCiJoXkFG5T0Sy8hK-aggH_rK2wrhIvUjV0PM2U_k5Ugs7mRklJlrP
  7SZcDqdqBtD4ODlm2TD1ywpNp09Ia6cFusYvtIgrGJPSJEITBXDZC-Iyy3zpuUb_HL_J4prtrWDVNjSUotv_k3
  qiZpSTFtIevs7ALfGDnik36QLchkQIbsouQ7iCLlzEcX6UIebCQ7FDMxvklpDq-3GKJyD4um2Zi7ColXSi9Sou
  LdnePJe04507b6ZaO7ByNvp04dXnLLgKwm4F6Cjl4TrbgPDyRXQwgWEJfY0a6ZCouZog-zVG6NS3ZD43UON1Hh
  9bITpfxecjwSRQtrD9TDI2xJJ4UEG5486KUG8bgOxzWNROQyU3wUkFAqYVpC3d1xp4B-iDGjv6QL1d6LyKxcjv
  2CENg8Iyg6j57Dip4eorGeDvWZVuqIRGAHA3Eobk1FBQPafZgTgg_ZGn_Qu8wdzdxP6U33q_XGB9jvN-ujltQ4
  qC7p4mZUA1m7AWypos-VL0CEMVOiT_O5EnWVzTwu5Wsq1AZRRyka0qPLYrl-o7SnOidQrj1Y2TMt3Ahmw58l57
  BYfSTRECpyEuuZ6qwj0Rvn79U9a8rZLxJVUbXLmxg3mcl4LkN3RYduUhbSb98eNH72EKbFQcUVtZIAD3pjlarm
  WWBgtNkuiv_4knvzejtr-bO4ZPJ-QQDueZDoT7q6Sxs4CVNuRMw9BSQ7W4uP-L-6UJBU6GeDlR0tTO7jkdcrCc
  XbMg7pXj5jN8YHO4M0oCc3o_601kB_Oh0jRv1KyOGa2HTS2jBG6R2Z0aTWS2b48tXk3pF-aZvK06QA22LkUM8e
  iuUNeXQYQzehEZ11cFRcSwAqDYCfvvJ07Jj7R4JBIv.EFPXR1G5W70Q-2GEMLkkqA"
```

```
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
    Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 508
      Version: TLS 1.2 (0x0303)
      Random: 856fae0da79967b8367ed3c895653c165d1e5d7bac34448950c70d884e29dc6e
      Session ID Length: 32
      Session ID: 88c8c53bfefe28037b1b38ed8af5078308e2290504fb9615ac2f8edb8aebb304
      Cipher Suites Length: 32
      Cipher Suites (16 suites)
        Cipher Suite: Reserved (GREASE) (0x4a4a)
        Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
        Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
        Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
```

(a) Card info (b) TLS protocol

Figure 2

4 Question3

Looking at the HTML you have saved, and if necessary any JavaScript etc. this calls, do you feel confident you know what it is doing with your data?

I am pretty sure what it did with my data. During the whole process, since I have logged in first, this website could get my contact information (first name and last name) from my profile. Then I inputted shipping info including 'postcode' and 'telephone number', and the billing address is the same as the shipping address, at last, the most important card info.
So they are all my data on this page. For the shipping info, it is only used in entry NO.61, the real 'purchasing' process, because if the payment is successful, shipping info would be used to create a new order.
While for the card info, referring back to entry NO.43(cybersource, 2022b), the card info is already encrypted before the browser tries to send it to another site (https://flex.cybersource.com/flex/v2/tokens) by flex.js(Microform, 2018), and it is well protected.
I found that 'Flex Microform is a CyberSource-hosted page that replaces the card number input field and calls the Flex API on your behalf.'(Microform, 2018). This js file enables the website to replace the sensitive card number input field with a secure iframe (hosted by CyberSource).
Instead of plain-text form, the Flex Microform would create a 'token' to represent the card info, which is mathematically irreversible and encrypted. And the 'token' can be used only by the customer for the following transactions in existing CyberSource APIs. Compared to functions using a plain-text form, this would be much safe

5 Question4

How dependent is the HTML/JavaSscript...you have in your example on the correct functioning of the DNS (Note that I am not asking how the DNS might be hacked, but rather what if.)? In particular, could bad DNS results (i.e., at any point, an adversary arranges for you to get incorrect, presumably to the adversary's advantage, DNS results.) result in a security problem?

The transaction's successful operation depends on the correct functions on the main sites, shown in Figure 1. In my case, see (Figure1), there are two situations that should be considered, since they are using important information 'card info' and 'shipping info'. There are:

- (a).STEP 1 : the DNS of cybersource(cybersource, 2022a) has been hacked, then my card info will be sent to the bad site.
- (b).Entry NO.61, right before STEP5 : the data including 'cartId', 'token' and shipping info will be sent to the bad site.

First of all, since all the 'POST' requests are using the https protocol and protected by TLS protocol, which can be evidenced in the Wireshark file. When the bad site gets the 'package', how to open it would be the first resistance. Then if they can open the 'package' by breaking the protection of the Cipher suite, they could see the request data.

In case (a), after they open the 'package', the only thing they get is the encrypted string 'eyJraWQi...Q-2GEMLkkqA', which is produced by flex(Microform, 2018). So it will be a new tough mission for the hackers, I do not know the encryption algorithm of flex, but I would believe it is not easy to be deciphered. Then the most important information card info will be quite safe and hope no financial loss would happen to the shopper. However, the response 'token' is the key parameter for the whole transaction, see the 'STEP 1' in (Figure 1), if the DNS CyberSource is hacked and gives nothing back, there would be no 'token' for continuing the transaction process. Then all the transactions on the site will not be accomplished, all the shoppers cannot buy goods, which is a huge loss for the online store.
In case (b), the bad site still has to figure out the 'package opening' mission. However, if the hackers deciphered the package, the data inside there is in plain-text form. Therefore, the shipping info including 'postcode', 'telephone number', and other private information inputted on that page would be seen by hackers. This might not cause financial loss directly, but if they collect them for other bad things, it would still lead to other security problems. As for the transaction, 'STEP 5' will never be executed since the online store site will not get the 'purchasing' signal, which would also cause financial loss for the online store as in case (a).

6 Question5

What makes you think that the sum of money displayed to you is the sum that will be transmitted to your bank?

By checking the logs, we can see right after the 'payment failure' entry NO.61, there are three entries with 'high' priority, i.e. entries NO.62, NO.63, and NO.64. Basically, they are doing the refresh thing, after a 'payment failure'. Entry NO.63 and NO.64 return data in JSON format, and NO.63 returns information about the prices like 'grand_total', 'discount_amount', 'shipping_amount', and so on, meanwhile NO.64 gives the data about the 'cart' which contains information like 'item_count', details about each item, and others.
All in all, they help keep data displayed in line with the data stored in the back end. Because the sum of money displayed on the website, is not calculated locally, it is bonded with the 'cart' using 'cartId' and the calculation happens in the back end, the front end only in charge of displaying. And from the 'postData' of entry NO.61, I think the important parameters are 'cartId', 'sessionId', and the 'token', combining them to make sure this transaction is dealing with this cart which is stored in the back end. So in my case, since data in the back end would not be attacked easily when the back end calls the 'purchasing function' by communicating with the payment service provider (Cardinal, 2022), the sum is the real sum from the back end database. So even if the browser has something wrong with price-displaying, as long as the 'cartId', 'sessionId', and the 'token' are correct, the sum will be transmitted to my bank will be correct.

7 Conclusion

What have you learned about the security of your card data?

I have learned the credit card online payment process, there are terms like 'Merchant account', 'Issuer', 'Card association', 'Acquirer', 'Payment processor', 'Payment gateway'(GOCardless, 2020). While in the (Figure1), I am pretty sure 'cardinalCommerce'(Cardinal, 2022) is the 'Payment processor', but I can not find out the 'gateway' is provided by cardinalCommerce or cybersource(cybersource, 2022a) based on what I got.
Therefore, the browser gives the card info that is encrypted by flex(Microform, 2018) and CyberSource to the 'merchant', then the 'merchant' securely transfers the transaction information including card info and the amount that should be paid to the payment gateway, and payment gateway securely transfers the transaction information to the payment processor 'cardinalCommerce'. After that, all the things would be done by cardinalCommerce, as it is a platform that integrated merchants, issuers, and others.
So, I think among all these stages, the data transition from browser to the merchant is the most insecure one, but this site integrated flex (Microform, 2018) and Cybersource (cybersource, 2022b) service, which made card data safer. And the PCI DSS(Council, 2020) requires that 'Merchants, service providers, and other entities involved with payment card processing must never store sensitive authentication data after authorization.' So I believe if my card data only be used by these trusted organizations, and well protected in the browser-merchant transition, it is safe.

In particular, what did you learn from the logs/HTML/etc. that you could not have reasonably deduced as a shopper with no access to these?

As a shopper with no logs, on this page, there is information on my account profile because I logged in first. So behind the ’screen’, there is some authorization with my account info, and by clicking some buttons I can go back to the homepage, which makes me trust this site. After inputting the card number, the icon image would change into ’Visa’, ’MasterCard’, which means it has the function to recognize card type. And there is some information like ’PayPal’, ’Buy now pay later’ Klarna and so on, therefore, from a normal shopper’s view, they are all the things I learned.
Sorry for not understanding this question well.

How obvious is the security of the website you used (and any services it calls) to the shopper?

As a shopper, if he/she is unable to capture the logs, I do not think he/she could find the security of the sites and services, opening this page from an official homepage would make him/her trust the site. But if he/she could get logs/HTML as I did, it would be capable to gain more information. There are 5 IP addresses the browser communicated during the whole process:

- 104.22.62.160 (https://store.liverpoolfc.com)
  Firstly, it is the official site of the merchant, then from Wireshark logs, it follows TLSv1.2 protocol when sending data from browser, from HAR logs, all the URLs start with ’https’, so I think the security of this site is quite obvious.

- 52.224.31.34 (https://h.clarity.ms)
  This is the Clarity from Microsoft(MS, 2022), these request URLs all start with ’https’, inside the ’postdata’ is encrypted string ’Åÿ7Ç_E-0ÉE3ÓüM:çx3àÝdc...’. From Wireshark logs, it is also following TLSv1.2, and I tested this function, it showed that this service captures actions like ’mouse pointer moving’. So I think the security of this service is also quite obvious.

- 18.170.126.184 (https://p.yotpo.com)
  The browser only uses ’GET’ requests to get images from this site, which means no data sending out, following HTTPS and TLSv1.2 protocol also. So the security of this site is obvious.

- 104.18.250.34 (https://flex.cybersource.com)
  This site occurred once only, the browser passed encrypted card info by flex(Microform, 2018), following HTTPS and TLSv1.2 protocol. Its security is obvious.

- 198.217.251.250 (https://xxxxx.cardinalcommerce.com)
  This site is the ’Payment processor’, request URLs all start with ’https’, following TLSv1.2 protocol in data transition. The browser did send order detail (without card info) to it, and got some js files like ’songBird’ to arrange the communications between the front-end and the ’Payment processor’. And this site provided ’JWT’ to the front-end, so I believe security of this site is obvious.

- 104.16.18.94 (https://cdnjs.cloudflare.com)
  Two requests with this site, they are ’GET’ with JSON format string in return. Google said CDN(Cloudflare, 2022) is a network service provider, but from my point, I can not say its security is obvious, since I did get enough information about it.

References

Adobe, 2021. *Cardinalcommerce 3-d secure*. https://devdocs.magento.com/guides/v2.4/payments-integrations/cardinal/cardinal.html.

Cardinal, 2022. *Cardinal*. https://www.cardinalcommerce.com/.

Cloudflare, 2022. *Cloudflare cdn*. https://www.cloudflare.com/cdn/.

Council, P.S.S., 2020. *Pci dss quick reference guide:understanding the payment card industry data security standard version 3.2.1*. https://www.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf?agreement=true\&time=1649376436722.

CWE-200, 2021. *Exposure of sensitive information to an unauthorized actor*. https://cwe.mitre.org/data/definitions/200.html.

cybersource, 2022a. *Accepting payments has never been easier*. https://www.cybersource.com/en-gb.html.

cybersource, 2022b. *Tokenize card*. https://developer.cybersource.com/api/soap-developer-guides/dita-flex/SAFlexibleToken/FlexAPI/tokenize_card.html\#Id18A3DI00ME9_Id18A3DL0304Y.

GOCardless, 2020. *How do online payments via credit or debit card work?* https://gocardless.com/guides/online-payments-guide/online-payments-credit-debit-card/.

Microform, F., 2018. *Flex microform*. https://developer.cybersource.com/api/soap-developer-guides/dita-flex/SAFlexibleToken/FlexMicroform.html.

MS, 2022. *See what your users want—with clarity*. https://clarity.microsoft.com/.

Simone, N., 2020. *Getting started*. https://cardinaldocs.atlassian.net/wiki/spaces/CC/pages/131806/Getting+Started.

TLS, 2022. *Transport layer security (tls)*. https://www.geeksforgeeks.org/transport-layer-security-tls/.

yopto, 2022. *Great brands are built on happy customers*. https://www.yotpo.com/.