

Security Analysis on Snapchat

Czarina Lao, Cheahuychou Mao, Adrian Sy

Abstract

Snapchat is a popular social media application that allows users to share media that are only stored for limited amounts of time. As such, we focused on network-based attacks on Snapchat's web and mobile applications, as well as their third-party integrations. In addition, we also analyzed the app code and the generated databases. Using various proxies and SSL bypass mechanisms, we were able to intercept sensitive data such as a user's password and account history. We also discovered network vulnerabilities for integrated applications that allowed attackers to observe user activities. Lastly, we were able to discover locally stored data from the Android app's databases, that are not normally visible when using the app.

1. Introduction

Snapchat is a multimedia messaging mobile application with over 150 million users worldwide. [18] Unlike other messaging applications, Snapchat allows users to share photos, videos and messages that get automatically deleted. As the app goes through rapid development, the security of the app has inevitably become harder to maintain. This project analyzes the security of the Snapchat mobile apps and web app to locate possible vulnerabilities that may have emerged in recent years due to new features and reliance on third-party applications.

2. Responsible Disclosure

The course staff has received permission from Snapchat Security Team for us to work on this project. Upon completion of the project, we will send the report to Snapchat. This report will be released shortly after that to allow Snapchat to fix any vulnerabilities we have identified.

3. Security Policy

According to the Privacy Policy (last modified January 10, 2017) from Snap, Inc.'s website, below are the principals and features of Snapchat, and the associated security policies. [23]

Snap, Inc.

- deletes Snaps from servers once all Snapchat users it was sent to have viewed it, unless legally required to do otherwise
- can keep data that was posted to a user's Story, Live, Local, and other crowd-sourced services for a longer period of time

Snapchat Users

- can add friends by username, from contacts list, by Snapcode, using Nearby, and using Quick Add which uses mutual friends
- can view, send, and save Snaps and Chats only sent to them and those in My Story of Snapchatters they have added
- can see username, name, Bitmoji, Snapchat score, and mutual friends of other users
- can view any additional information that a user has consented to share, including whether someone has a Snapchat account based on phone numbers from a user's phonebook
- cannot view deleted accounts, even if account information is still in server (for 30 days after deletion)
- cannot send Snaps and Chats and view the Story of users who have blocked them

Everyone

- public information of any user such as names, username, Snapcode, and profile pictures
- any content submitted to crowd-sourced features like Live and Local

Third-Party Advertisers

- can include attachments to ads, which include articles, app installations, long-form videos, and website all within the Snapchat app
- can use cookies, web beacons, and similar tracking technologies
- can target Snapchatters by age, gender, geographical location, mobile device, operating system (Android or iOS), mobile carrier, interests, Snapchat's first-party audiences, lookalikes, and purchase intent with Datalogix segments

Features

- Memories - has access to phone camera roll photos

- Snapcash - uses debit card information and sent by using a \$ in Chat
- Geofilters - have access to location, can be available only to a certain Geofence (bounded area) for a certain period of time
- Groups - can send Snaps and Chats to up to 16 users at a time
- Unlockables - can analyze Snaps taken (even if they're not yet sent) for any Snapcodes
- Shazam - can detect and analyze background audio
- Other Third-Party Integrations - can get any information (including personal information) based on the Terms and Services that they provide

4. Related Work

Snapchat security vulnerabilities have been explored and revealed over the last few years by hackers, security experts, and regular users. Feasible attacks have been denial-of-service attacks, spoofing attacks, man-in-the-middle attacks and social attacks. Snapchat has been very careful and swift in fixing vulnerabilities that have been revealed. However, these findings have led to a lot of interesting questions about the security and privacy that Snapchat seeks to provide to its users.

4.1 Security Threats in Late 2013

Gibson Security did an intensive security analysis of the Snapchat Android API back in December 2013 and released a full disclosure report that unveiled numerous concerning security flaws in the API. 4.6 million usernames and phone numbers were leaked because Snapchat used a hardcoded encryption key for all accounts which enabled an adversary to save the media sent through Snaps and build such a database of user information. [6]

4.2 Third-party Apps and Plugins

Third-party plugins and apps provide users with ways to circumvent Snapchat's content viewing and other restrictions. Popular ones include Snap Upload, Casper, Snap Crack, Quick Upload, Snapchat++ and Phantom. [24] Previously, on iOS, these tweaked Snapchat apps could only be used on jailbroken devices, but Apple now allows the sideloading of apps. As such, with a sideloaded app called TweakBox, iOS users can now install tweaked Snapchat apps. Snapchat itself is very aware of the vulnerabilities that come with those tweaks, and explicitly warns its users not to install any of the plugins and apps as it believes that those can compromise the

security of one's account and the accounts of one's friends. [24] In the past few months, Snapchat accounts were locked/suspended for using such third-party apps. We explored methods to obtain the same functionality provided by third-party apps in order to circumvent Snapchat's DRM (Digital Rights Management) monitoring.

4.3 User-found Security Holes

Snapchat users have managed to find ways to break the security and privacy restrictions, for example, turning the phone to airplane mode between loading Snapchat messages and opening the message so that one can take a screenshot of Snapchat message without notifying the sender or view the message for longer than supposed to. [10] It is also discovered that one can change the date on his phone to bring back filters offered on some days in the past. [28] So, a user could use this to trick his friends with regards to when the photos were taken, which is a form of social attack. However, most of these have been patched by Snapchat as of the latest update.

4.4 Previous Student Work

Last year, past 6.857 students did a security analysis of Snapchat and performed a number of attacks on both the Snapchat iOS and Android app. Under certain conditions, they were able to expose sensitive user information and violate content viewing restrictions. In particular, they were able to obtain root access on an Android device, and therefore gain access to the database that includes sensitive information that is normally not visible to users. However, despite successfully obtaining the Android APK and decompiling it, the students were unable to recompile and run Snapchat after editing its source code. In addition, for the web application, they managed to intercept packets sent between `snapchat.com` and `accounts.snapchat.com`, edit the packet headers, and replay both POST and GET requests. The students also found that unverified users were able to access part of `geofilters.snapchat.com` that only verified users should. [11]

5. Summary of Attacks

To test the security measures used by Snapchat, we attempted a variety of network-based attacks against Snapchat's mobile applications as well as its web application. In addition, we

examined applications that were recently integrated into Snapchat. We also examined the application's code and the generated databases for confidential data.

5.1 Network Attacks

Snapchat uses SSL (Secure Sockets Layer) / TLS (Transport Security Layer) to provide privacy and data integrity for its data communication. All data transmitted over the internet are encrypted using symmetric key encryption and ensured integrity with message authentication codes. In addition, the identity of the communicating parties is authenticated using public key encryption. The protocols rely on trusted third-party certificate authorities to establish the authenticity of digital certificates, which are typically used as a proof for ownership of the public key. [5] This reliance on certificates authorities, however, is known to leave TLS/SSL susceptible to the man-in-the-middle attacks by attackers who have certificates signed by an authority the client trusts. [19, 20] Under the consideration that users might get tricked to trust malicious certificate authorities and that all security guarantees are only valid when HTTPS is used, we installed a couple of HTTPS proxies on a number of machines and devices to simulate attacks that adversaries could perform.

The HTTPS proxies used are Charles and mitmproxy. [4, 12] If configured correctly as described in detail below, the proxies can capture the traffic between the user and the Snapchat web servers and provide plaintext view of the traffic. One also has the option to modify, replay and abort packets. However, by using an SSL proxy, we discovered that in some cases, Snapchat uses SSL pinning or HTTP Strict Transport Security (HSTS) as another layer of protection to prevent man-in-the-middle attacks. This layer ensures that trusted certificates are correct according to the browser or OS itself, and that the users are always on HTTPS. [7] Therefore, many attacks are not feasible without further efforts. The sections below describe the successful and failed attacks that we performed.

5.1.1 Web Application

Snapchat provides a web interface (`accounts.snapchat.com`) solely for account management purposes. Users may login to download or generate new Snapcodes, download in-depth account history, change one's password, and unlock or delete one's account. The students from last year examined the vulnerabilities against cross-site scripting attacks, SQL injection, cross-site request forgery, brute force login and source inspection, and concluded that

it was secure. [11] As such, we decided to explore attacks related to sensitive information that attackers could get from this web interface.

Vulnerabilities of Different Browsers

We first used Charles as an HTTP proxy to sniff the traffic without trusting the Charles root certificate or enabling SSL proxy. The observed traffic was incomprehensible. After enabling SSL proxy, Chrome blocked us from accessing the website, warning that the connection was not private because the credentials were incorrect, and that due to HSTS, we are not allowed to visit the website. Firefox (52.0.2) and Safari (9.1.2), on the other hand, showed us a warning that it could not verify the identity of the website but still gave us the option to proceed by accepting the certificate. We consider this as a potential vulnerability because there are users out there who are not aware of the risks of proceeding.

After proceeding, we were able to login after completing a reCAPTCHA check. For both Firefox and Safari, we were able to observe all the traffic in plain text and user friendly formats. As shown in Figure 1, that includes the username and password the user enters at login, the Snapcode and more, as the user navigates through the web app. We were able to intercept, modify, replay and abort both the request and response packets.

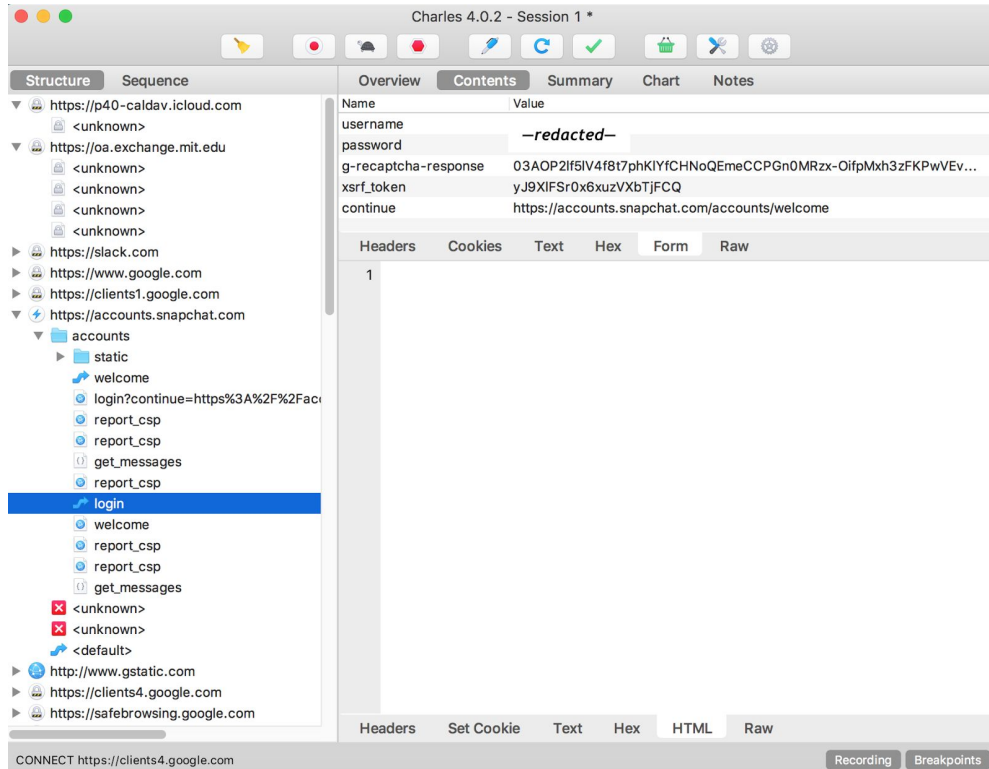


Figure 1: An intercepted packet containing the user's username and password

Deleting a User's Account

Snapchat requires users to fill in their username and password to confirm account deletion. The form, in addition, contains an xsrf token to protect against cross-site request forgery (XSRF) attacks. However, the form does not contain a reCAPTCHA puzzle and the xsrf token is identical to that of all forms for the entire login session. With the HTML that Charles and mitmproxy provided us, after the user logs in, we were able to forge a POST request using the xsrf token, username and password to delete the account. This can be done without the user having opened the Delete My Account page. It can also be automated due to the absence of reCAPTCHA puzzles, and the user can be redirected to the login page to avoid suspicion. The user will get a notification email, but there is no further notification from the mobile app; instead, it silently logs the user out. The user has 30 days to recover the account without any loss of information by simply logging back in. However, it could take a while for the user to realize that since the user cannot receive any Snaps or money transactions once the account is deleted.

Changing a User's Password

Using a similar method, we were able to change a user's password through a man in the middle attack. Likewise, the user is logged out of the Snapchat mobile app after the password change. The user is only notified via email and will not be able to undo actions done by the attacker using the stolen account, e.g. sending Snapcash to the attacker's account. There is, however, the option to recover one's password since Snapchat explicitly prevents username changes as a security measure.

Advanced Account Attacks

Given the abilities described above, an attacker could permanently block a user from his/her account by deleting the account after changing the password. The user will not be able to recover the account by logging into the app on their device because the password has already been changed. Password recovery will not work because the account with the original username and password combination simply does not exist. Similarly, due to the absence of a reCAPTCHA puzzle for these actions, the attackers will be able to automate this and perform this attack on a large scale. Alternatively, the attacker could manually changing the email address of the user on the app before changing the password. This effectively transfers the account to the attacker without notifying the user. The attacker could then impersonate the user and steal money from the user's credit card by using it to pay his/her personal account.

Stealing Account Activity History

The Snapchat web app allows users with verified email addresses to download very detailed data about their app usage history including geographic locations, the machines and IP addresses used to login, friend information, Snap history (sender, receiver, timestamp) and purchase history. To download this information, a user simply goes to My Data page and clicks on Submit Request. The user is notified via email within a few seconds that the data is ready. He/she can refresh the page to see the link for downloading the zip file that contains the HTML files with all the information. As an attacker, we were able to steal the link and download the zip file from a different machine since Snapchat does not check for any form of authentication. This is again another exploitable security flaw because attackers could use this information for malicious purposes such as tracking the user.

Account History and Information

Basic Information

Username: --redacted--

Name: Test 857

Creation Date: 2017-04-23 06:25:15 UTC

Privacy Policy and Terms of Service Acceptance History

Effective Date/Name: 2016-07-06 UTC
Acceptance Date: 2017-04-23 06:28:35 UTC

Effective Date/Name: 2017-01-10 UTC
Acceptance Date: 2017-04-23 06:25:16 UTC

Login History

IP: --redacted--
Country: US
Created: 2017-05-15 00:55:36 UTC
Status: success
Device: --redacted--

Figure 2: The account history provided by the web app

Replaying and Aborting Packets

Another attack that we performed was dropping packets to prevent the user from using the site. While this might not be a useful attack for a lot of cases, it could be quite troublesome if the user lost his/her phone and needs to delete the account or change the password to log himself/herself out of the device. Lastly, we also did packet replay. However, we discovered that each replay could only last for 10 minutes because the cookies expired after that. This might be one way that Snapchat prevents denial-of-service attacks.

5.1.2 Mobile Applications

Snapchat transfers and keeps track of a lot of user information such as geolocations, chats, images and videos. To this end, confidentiality and data integrity are no doubt crucial. Over the last few years, Snapchat had integrated numerous features of partner apps into the app. Those include the peer-to-peer payment and in-app purchases supported by Square, video chat service provided by AddLive, personal emoji stickers in messages provided by Bitmoji and music recognition by Shazam. Hence, the security of its mobile application has inevitably become dependent on the security of those partner apps. The next sections describe how we

used Charles and mitmproxy to observe and interfere with the network traffic to explore the vulnerabilities that could manifest as a result of these integrations.

SSL Pinning

Snapchat does SSL pinning for both Android and iOS device: the app saves the X.509 certificate unique to each device to detect user attempts to access the app using a different certificate. [15] When the user opens the app on both Android and iOS devices, it sends a CONNECT method request to `app.snapchat.com` to establish a tunnel for the SSL connection. [13] Therefore, when we had the traffic configured to go through Charles Proxy or mitmproxy, we were unable to use the Snapchat app at all.

On iOS devices, we were able to bypass this by using SSL Kill Switch. We used an iPad running iOS 9.0 which was jailbroken using Pangu, and sideloaded SSL Kill Switch 2. [17, 2] With SSL Kill Switch turned on, we were able to observe all traffic through the app and perform man-in-the-middle attacks using Charles Proxy.

However, we were unable use the Android version of SSL Kill Switch on a rooted Galaxy S4 running Android 5.0.1 due to version incompatibilities. Although we were unable to observe traffic on the Android app, it is safe to assume that the data transfer protocols are very similar to that of the iOS version. We decided to ignore the traffic to the Snapchat domain and explore other related traffic instead. The app was completely functional under that configuration, and we were able to observe some traffic related to features that are supported by partner apps that the Snapchat app uses.

The third-party integrations, Bitmoji and Shazam, however, do not use SSL pinning. As such we did not have to use a rooted/jailbroken device in order to access their traffic.

Stealing Bitmoji Avatar

Bitmoji allows users to create an expressive cartoon avatar featuring themselves and obtain a huge selection of such stickers to use in multimedia messages. Using Bitmoji in Snapchat unlocks friendmoji which features the user and the friends the user is chatting to. To login to the Bitmoji app using a Snapchat account, users are redirected and required to login through the

Snapchat app. With SSL proxy disabled for the Snapchat domain and enabled for the Bitmoji and Bitstrips domains, we were able to login to the Bitmoji app and observe the traffic through Bitmoji on Charles Proxy and mitmproxy. We were hoping to observe some transfer of credential information such as during the user login. However, all we observed was an access token that Snapchat provides to the user in order to use Bitmoji app.

Nonetheless, as the user browses through his/her Bitmoji stickers and settings in Bitmoji app, we were able to see all the stickers and outfits for the avatar, together with a json that contains detailed information about the avatar, notably, the avatar ID and hashed avatar ID that are unique to the account. When the user recreates a new Bitmoji, as expected, we were able to see all the actions and similarly steal the stickers for the new avatar.

We note that creating a Bitmoji avatar through Snapchat does not make a Bitmoji account for the user. As such, in order to modify one's avatar through the Bitmoji app, a user would have to go through the Snapchat app's login. However, users have to the option to create a Bitmoji account and link their created avatar through Snapchat. Users who do so will then be able to use their Bitmoji account to log in to the Bitmoji app and perform the aforementioned actions on their avatar. We discovered that brute force guessing of passwords in Bitmoji is possible, so if a Bitmoji account is compromised, the attacker could modify/delete a Snapchat user's Bitmoji avatar. Compromising a Bitmoji account is more likely since the Bitmoji app does not use SSL pinning.

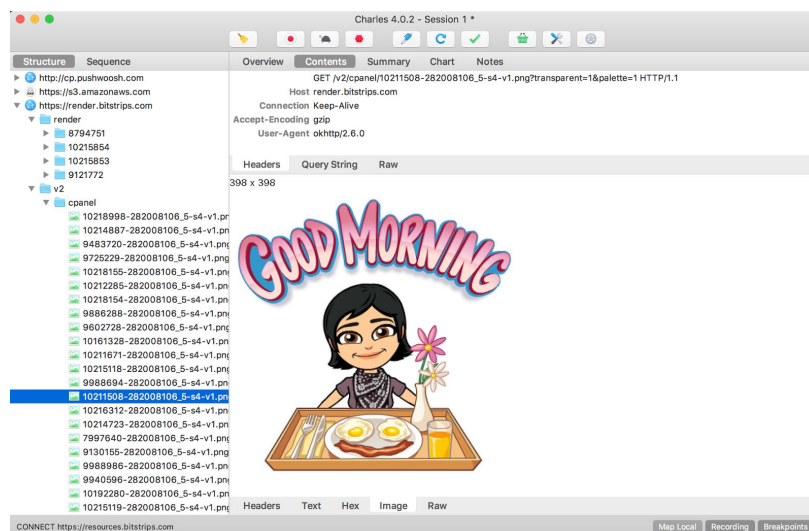


Figure 3: Intercepted Bitmoji sticker

Using Shazam to Track User Music Interests

Shazam is an app that can identify music based on a short sample played and using the microphone of the device. The integration into Snapchat allows users to send the music they are listening to and artist discoveries as Snaps to their friends. Through Charles Proxy, we saw that for both Android and iOS, Snapchat made a POST request to Shazam to identify the music and Shazam responded with JSON that contains metadata for the song and some key and URL to the song. When the user clicks on More Info, there are JSONs for the lyrics and metadata of other related or recommended songs. We were however unable to observe any traffic when the song information gets shared to a friend via Snaps. Despite these, this could still lead to a privacy issue since the attackers will be able to know exactly what the user is listening to. We attempted to capture the audio that gets sent through the POST request since if an adversary could do that, he/she could use it to spy on Snapchat users. However, we could not retrieve it using our limited knowledge of streaming and VOIP.

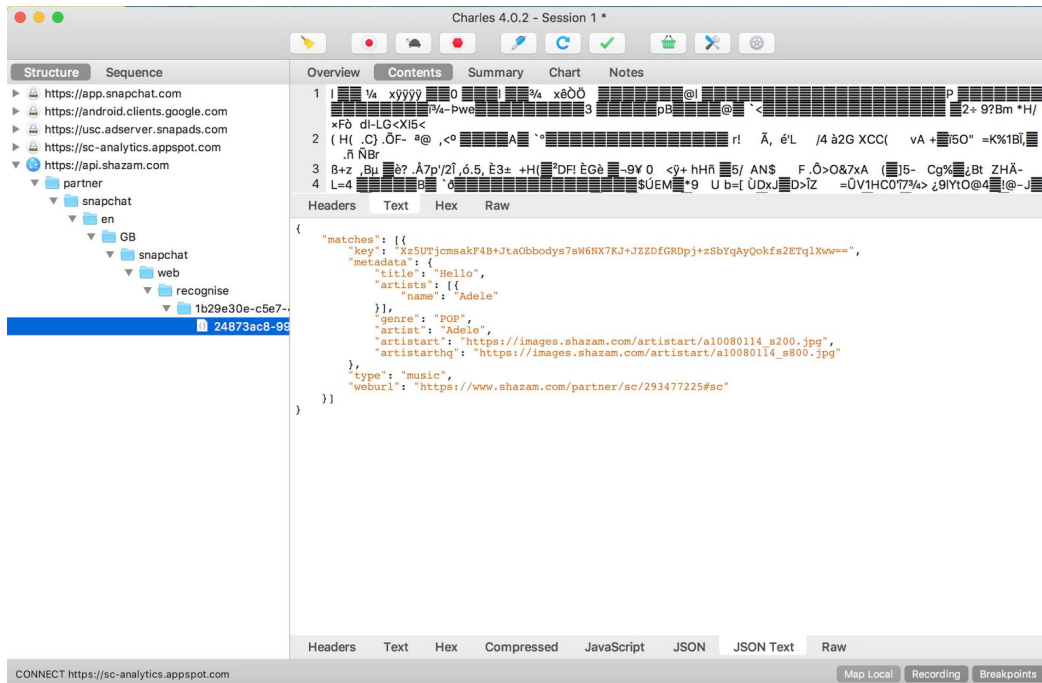


Figure 4: Intercepted Shazam packet

Snapcash

Snapcash is an in-app money exchange service meant to enable users to quickly pay each other for restaurant bills or concert tickets. [25] Users pay each other by simply entering a dollar

sign and the number of the amount to pay in the chat box and send it to the receiver. The transaction is then processed by the Square API. It is likely that Square uses SSL pinning for both Android and iOS as observed from network traffic. Moreover, we did not attempt attacks involving Square since we do not have permission from the company to do so. As a result, we were unable to do much testing or analysis. We did try out simpler attacks such as entering a negative number. However, Snapchat simply disallowed that. So we conclude that under the assumption of Snapchat's security, Snapcash is also secure.

The attacks below were performed only on a jailbroken iOS device. However, we expect that the attacks are equally feasible on Android devices that bypass SSL pinning successfully.

Location Tracking

When taking a Snap, the Snapchat app makes a POST request that contains the latitude (lat) and the longitude (long) given by the location of the device. Passive man-in-the-middle attackers could observe all these POST requests and track the physical location of compromised users.

Location Spoofing

When the Snapchat app makes the POST request that contains the location of the device, as described above, it receives a response containing links to the applicable filters and lenses for the user. We were able to spoof the user's location by writing a rule that intercepts all such POST requests and replacing the lat and long values in the request. With location spoofing, users get geofilters that are not in their actual location. We were able to get the same feature without using a tweaked app, such as Phantom, which last year's students had to use. [11]

Exploiting Google Storage Files

Many of the responses to the requests that the Snapchat app makes return JSON data containing information such as links to Google Storage data. As a passive listener, we were able to get the links from the responses and access the links through a web browser. We had varying degrees of success as listed below:

- **Stickers, Filters, Lenses, Logos, and Thumbnails:** We were successful in downloading all these files (See Figures 5). Stickers and lenses came in ZIP files, while the filters, logos, Story thumbnails, and lens thumbnails came as PNG files.
- **Stories:** We were able to download Story binary files, but we were unable to open/decrypt them.
- **Memories:** We were unsuccessful in downloading anything from the URLs found and got the error message that Google Storage needed a valid signature.



Figure 5: A downloaded folder of all the Stickers in Snapchat

Response Modification

We were also able to modify the responses received from the HTTP requests, just as we were able to write rules to modify the POST requests for location spoofing. When the app first starts up, it sends a POST request to `app.snapchat.com/lens/v2/load_schedule`, then receives a JSON containing filter and lens data, including the URLs to the images, relevant files, and other details.

- **Lenses**

We successfully replaced the Google Storage URLs mentioned above with links of our choice. For example, Figure 6 shows a lens icon changed to a PNG file from a non-Google Storage link. We were also able to bring back lenses that were only available a few days before as shown in Figure 7 by using the links that we obtained from observing the network traffic that day. While the lenses have a signature field, the

original lens link could still be properly loaded even if the signature is changed. Similarly, the lens from the changed lens link was loaded properly without changing the signature field. We observed the signature field for a lens to remain the same whenever we restarted the app.

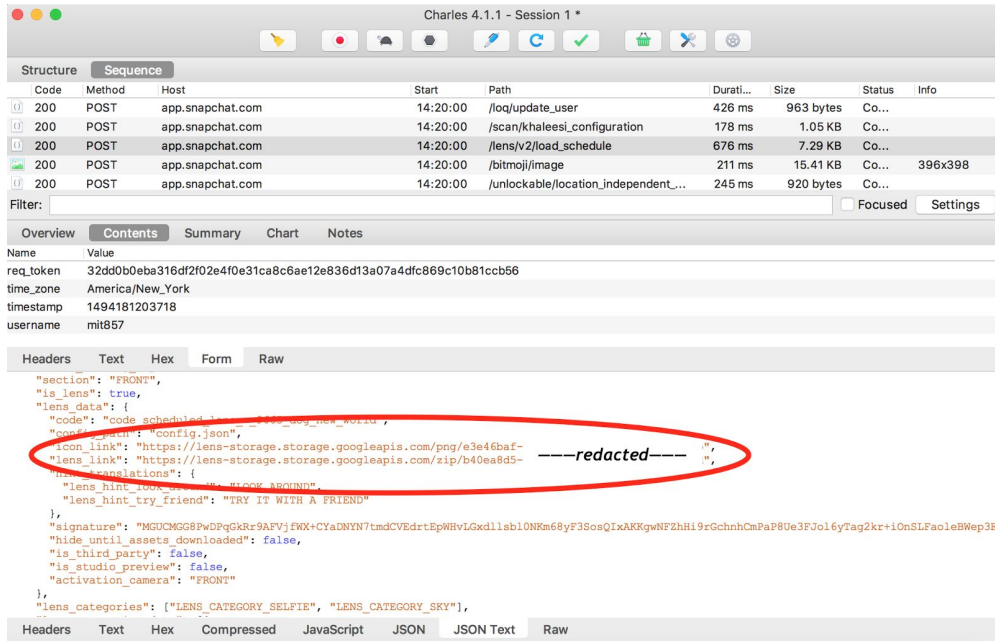


Figure 6: Lens data, including links to the lens files, in the JSON response on Charles



Figure 7: Modifying lens URLs. Using the lens from the MTV Movie & TV Awards from May 7, 2017 a week later

- Filters

Similar to the lenses, the links to filters can be changed to both a link to a previously available filter or a link to an outside source (Figure 8).



Figure 8: Modifying filter URLs. Using a custom-made geofilter from an outside link

- **Links to Malicious Sites**

Since we were able to modify the links that lenses and filters were downloaded from, we think we attempted changing the link to lead to a PDF file. By looking at the network traffic on Charles, we saw that the app did download the PDF file, but it was not properly displayed as a filter. Using a similar method, users can theoretically be made to unknowingly download malicious files.

Viewing Snaps

Each time a Snap is viewed, a POST request to the path `/update_snap` is sent to notify the server about the new action taken on the Snap. We tried to drop the requests to `/update_snap` but the sender of the Snap was still able to see that the receiver has viewed the snap. The said POST request also contains a “replayed” field which is set to 0 if it hasn’t been replayed and 1 if it has. We were able to change the field to 1 so that the sender/receiver of the Snap is mistakenly told that the Snap was replayed when it was not.

Viewing Chat Details

When we opened up Chat, the app sends a POST request to the path `/log/conversations` and receives a JSON of detailed Chat data. For instance, we were able to see the Shazam links a friend had previously sent through Chat. The links were no longer visible from the app, but they were still in the JSON sent. Group chat information are also available such as the participants of the chat and the group name. Actions in group chats can also be seen in the network traffic such as changing the group name.

Obtaining User Information

As expected, we were able to obtain detailed user information while viewing the plaintext network traffic made by the Snapchat app. At login, we were able to see the username and the password, as well as user device information, in the POST request made to `/log/login`. This request then gets a JSON response of even more information about the user. These include the user's user ID, Bitmoji avatar ID, email, birthday, number of sent and received Snaps, Snap score, various settings and preferences, and the name, user ID and Snap streak count of all the user's friends.

5.2 Code Analysis

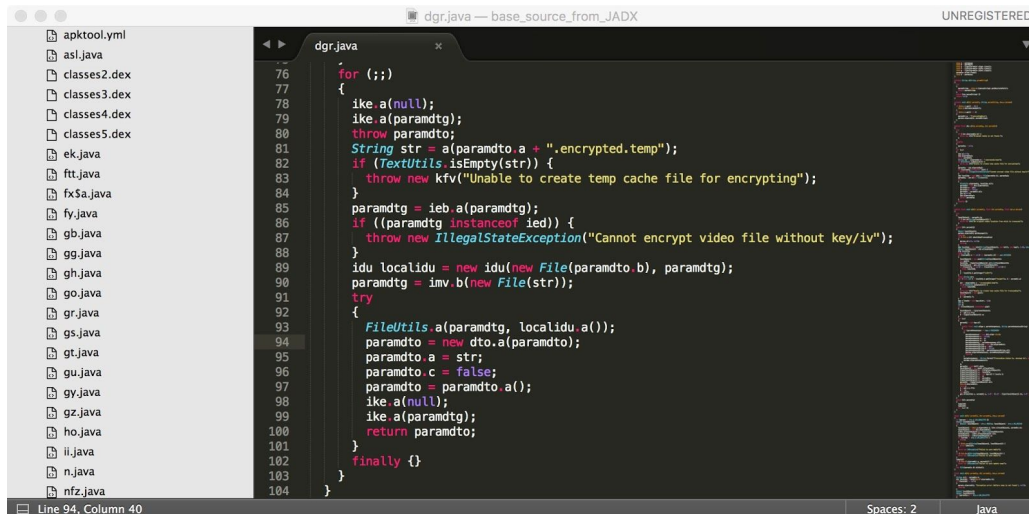
5.2.1 Android Application

For testing Snapchat's security for its Android application, we used a Samsung Galaxy S4 that was rooted using KingoRoot. [9] While Snapchat supposedly blocks devices with root access from logging-in, it seems like it only checks for certain kinds of bypasses because we were able to log-in without any problems. [22] Using a file explorer, we were able to find and extract the apk file of the application as well as other data files used by the application such as database files.

APK Analysis

We were able to get the apk file for the Snapchat Android application and decompile it into Java files using Android APK decompiler or Smali files using APKTool. [3, 14] However, the Java files and Smali files were obfuscated and we were not able to discover much about the code other than the fact that Snapchat encrypts videos using some encryption scheme that requires an initialization vector and a key as shown in Figure 9. Additionally, the hard-coded certificates and

developer endpoints found by the last year's students are no longer in the current version of the APK. [11] We were unable to recompile the APK with or without edits to the code.



```
76     for (;;)
77     {
78         ike a(null);
79         ike a(paramdtg);
80         throw paramdtg;
81         String str = a(paramdto a + ".encrypted.tmp");
82         if (TextUtils.isEmpty(str)) {
83             throw new kfv("Unable to create temp cache file for encrypting");
84         }
85         paramdtg = ieb a(paramdtg);
86         if ((paramdtg instanceof ied) {
87             throw new IllegalStateException("Cannot encrypt video file without key/iv");
88         }
89         idu localidu = new idu(new File(paramdto.b), paramdtg);
90         paramdtg = imv.b(new File(str));
91         try
92         {
93             FileUtils a(paramdtg, localidu a());
94             paramdto = new dto.a(paramdto);
95             paramdto.a = str;
96             paramdto.c = false;
97             paramdto = paramdto.a();
98             ike a(null);
99             ike a(paramdtg);
100            return paramdto;
101        }
102        finally {}
103    }
104 }
```

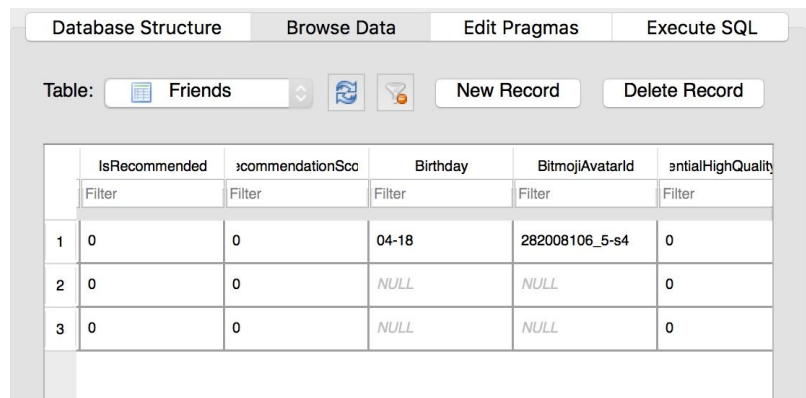
Figure 9: Obfuscated code referencing use of ivs and keys

Database Dump

The Android version of the Snapchat application stores structured data within SQL databases. These databases are likely used as caches to speed up loading times as compared to loading data from the servers. Normally, these database files are inaccessible to the user, but we were able to extract these files and export it to a laptop using a rooted phone. The files were located at `/data/data/com.snapchat.android/databases/tcspahn.db`. Using MySQLite to open the databases, we were able to access some data that would normally not be visible in the application.

Similar to the findings of last year's students, we found that most of the databases were empty with the exception of a few tables. [11] Among the non-empty tables, we were able to find a `Friends Table`, `FriendsWhoAddedMe Table`, and a `MediaCache Table`. The first table showed a list of all the user's friends and their birthdays which is not normally visible from the app itself as shown in Figure 7. Similarly, the second table showed how each friendship was formed (via Snapcode or username). Lastly, the `MediaCache` table contained paths to where encrypted saved images are stored along with a key, but we were unable to decrypt the images.

While none of the data found in the databases is explicitly confidential, attackers with access to these would be able to infer real world relationships of people based on when and how they added each other on Snapchat.



	IsRecommended	RecommendationScore	Birthday	BitmojiAvatarId	PartialHighQuality
	Filter	Filter	Filter	Filter	Filter
1	0	0	04-18	282008106_5-s4	0
2	0	0	NULL	NULL	0
3	0	0	NULL	NULL	0

Figure 10: Birthdays which are normally not visible from the app are stored

5.2.2 iOS Application

Last year's students analyzed the iOS app by obtaining the source code from a jailbroken device, decrypting it and using class-dump to produce the public and private methods defined. They then attempted to modify the code to bypass SSL pinning to reduce the dependency to jailbroken devices with SSL Kill Switch. [11] They were not successful, so instead, we decided to focus on going through the files to see what else we can obtain from them. While we found a few assets such as graphics and audio files, we were not able to gain anything significant from the iOS application IPA file that we unzipped. We did find a file, `certkey.pem`, which contained an RSA private key, but it was encrypted with DES-EDE3-CBC. The initialization value is also in the file which is standard. [26] We couldn't make use of it since we were not able to decrypt it without the passphrase needed to do so. Moreover, there are also various developer comments that contain links to company-internal tools in the files. Specifically, in the same file mentioned above, a link to a code review tool is present; however, when we tried to access the link, we were required to provide Snapchat login credentials.

5.3 Other Findings

Despite the app trying to resend the updates on the viewed/screenshotted/replayed Snaps to the server as a countermeasure to the method described in Section 4.3, we were still able to

take screenshots of Snaps without the sender knowing. To do this, we simply opened the Snap, turned off the network, took a screenshot, closed the app and cleared its cache. Deleting and reinstalling the app would have the same effect.

Furthermore, while combing through the application code for both iOS and Android we were able to find several developer notes within the code and links to their private Github account and their internal code review system. These sites were not accessible without proper authentication.

6. Recommendations

6.1 To Snapchat

6.1.1 Enforce HSTS

To protect users from network attacks, it is crucial that Snapchat web app only allow web browsers to connect to it through known certificates and HTTPS. Uninformed users might get tricked by adversaries to trust malicious certificate authorities that allow them to mount man-in-the-middle attacks and perform all malicious attacks described earlier.

6.1.2 Using Third-party Integrations

While the absence of SSL pinning only allows the adversaries to observe user activities, it would be better if no information were revealed. As such, we recommend that third-party applications also implement SSL pinning for additional security.

6.1.3 Protect Google Storage URLs

Although no personal or confidential data are leaked by making sticker, lens, filter and other files downloadable, we recommend employing the principle of least privilege. Since users do not need access to those files, it would be better to make the linked files inaccessible to anyone who does not need them. Moreover, the method to protect the Google Storage files is already used on the files for Memories, so it is be feasible to use them for other files as well.

6.1.4 Clean Up Production Code

In addition to obfuscating code, we recommend that comments for internal use also be removed from application code in both iOS and Android. Although we did not find any vulnerabilities from

the comments, there could be potential problems if the third-party sites in the links are compromised.

6.2 To Snapchat Users

Many of the points below are also emphasized by Snapchat, but we would like to reiterate them given how they relate to the results of our security analysis.

6.2.1 Verify Account Details

During account registration, Snapchat does not check whether or not the user enters a valid email address and simply allows the user to use the app without verifying the email address. While this is convenient and possibly an intentional design decision by Snapchat, we suggest that users provide and verify a valid email address for account recovery purposes.

6.2.2 Utilize Two-Factor Authentication

The attacks performed above rely on the fact the users only provides one form of authentication. If the user uses two-factor authentication, the attacker will not be able to use the stolen username and password to login to a different device and perform malicious actions such as stealing money through Snapcash.

6.2.3 Beware of Malicious Downloads

As discussed earlier, SSL relies on reliable certificate authorities and the use of HTTPS. Users should not trust any unknown certificate or agree to proceed to a website that uses only HTTP especially for web apps that involve sensitive information like Snapchat.

6.2.4. Do Not Tamper with your Device

While rooting or jailbreaking a device could give users greater flexibility and access to other features and applications, it exposes users to more security risks. Many apps including Snapchat explicitly warn their users against that. [21]

6.3 To Future Researchers

We would recommend to future researchers to test components that we were not able to cover because of various limitations. For one, we were not able to obtain the recently released Spectacles, a pair of camera-equipped sunglasses, that can connect to the Snapchat app. It would be worth analyzing the security of the device itself, as well as the data transfer between the Spectacles and the app. Moreover, if future researchers are able to get permission from Square, we recommend exploring the security of Snapcash further.

In addition, we would also recommend users with more experience in app development to further test the security of the app code and whether or not it would be possible to tweak the code to produce a functional malicious version of the app.

7. Conclusion

This project covered Snapchat's web application, iOS application and Android application, as well as recent app integrations. For mobile applications, we found that the app itself is fairly secure unless an attacker has heavily tampered with the device used to run the app. Similarly, the web application is only vulnerable when the attacker has set up a means to intercept traffic as well as bypass SSL on the device. Integrated applications, however, are not as secure because the user's activities on those applications can be exposed even on non-rooted/non-jailbroken devices. Nevertheless, we do note that no sensitive information such as passwords is revealed.

8. Acknowledgements

We would like to thank Professor Ron Rivest and the rest of the 6.857 staff for obtaining permission from Snapchat to perform this security analysis. We are also grateful to Snapchat, particularly, Dr. Moti Yung, who was the main point of contact on Snapchat's security team. Lastly, we thank our TA's Heeyoon Kim and Cheng Chen who provided helpful suggestions and guidance.

9. Bibliography

- [1] Akshay Gangwar. *How to Sideload Apps on iPhone in iOS 10 (Without Jailbreak)*. Feb 28, 2017. URL: <https://beebom.com/how-to-sideload-apps-iphone-ios-10-without-jailbreak/>

- [2] Alban Diquet. *SSL Kill Switch 2*. Feb 6, 2017. URL: <https://github.com/nabla-c0d3/ssl-kill-switch2/>
- [3] Apktool: a tool for reverse engineering Android apk files. URL: <http://ibotpeaches.github.io/Apktool>
- [4] Charles. *SSL Proxying*. URL: <https://www.charlesproxy.com/documentation/proxying/ssl-proxying/>
- [5] Digicert. What Is SSL (Secure Sockets Layer) and What Are SSL Certificates?. URL: <https://www.digicert.com/ssl.htm>
- [6] Gibson Security. *Snapchat - GibSec Full Disclosure*. Dec 23, 2013. URL: <http://gibsonsec.org/snapchat/fulldisclosure/>
- [7] *HTTP Strict Transport Security*. URL: <https://https.cio.gov/hsts/>
- [8] ios9cydia. *Jailbreak iOS 9*. URL: <http://www.ios9cydia.com/ios-9-jailbreak.html>
- [9] *Kingoroot*. URL: <https://www.kingoapp.com/>
- [10] Lewis Painter. *How to screenshot on Snapchat without them knowing*. Apr 7, 2017. URL: <http://www.pcadvisor.co.uk/how-to/social-networks/how-screenshot-on-snapchat-without-them-knowing-2017-3634217/>
- [11] Madeleine Dawson, Heeyoon Kim, Wei-En Lee, Kelly Shen. *Security Analysis of Snapchat*. May 2016. URL: <http://courses.csail.mit.edu/6.857/2016/files/11.pdf>
- [12] *mitmproxy*. URL: <https://mitmproxy.org>
- [13] Mozilla Developer Network. *HTTP Request Methods: CONNECT*. Feb 6, 2017. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT>
- [14] *Online Android decompiling tool*. URL: <http://www.decompileandroid.com>
- [15] OWASP. *Certificate and Public Key Pinning*. Jul 5, 2016. URL: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- [16] OWASP. *HTTP Strict Transport Security Cheat Sheet*. Jan 6, 2017. URL: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

- [17] Pangu. URL: <http://en.9.pangu.io/>
- [18] Recode. *Snap's IPO numbers look a lot more like Twitter's than Facebook's*. URL: <https://www.recode.net/2017/2/2/14492932/snapchat-ipo-twitter-facebook-comparison>
- [19] Scott Ogrin. *Why HTTPS and SSL are not as secure as you think*. URL: <https://www.sott.net/article/275524-Why-HTTPS-and-SSL-are-not-as-secure-as-you-think>
- [20] Security StackExchange. *How Safe is SSL*. URL: <https://security.stackexchange.com/questions/53596/how-safe-is-ssl>
- [21] Snap Inc. *For Snapchatters with Jailbroken iPhones*. URL: <https://support.snapchat.com/en-US/article/jailbroken-iphone>
- [22] Snap Inc. *I Can't Create An Account Or Access My Account*. URL: <https://support.snapchat.com/en-US/a/account-questions>
- [23] Snap Inc. *Privacy Policy*. Jan 10, 2017. URL: <https://www.snap.com/en-US/privacy/privacy-policy/>
- [24] Snap Inc. *Third-Party Applications and Plugins*. Apr 20, 2017. URL: <https://support.snapchat.com/en-US/a/third-party>
- [25] Snap Inc. *What Can Snapcash Be Used For?* URL: <https://support.snapchat.com/en-US/a/snapcash-guidelines>
- [26] Stackoverflow. *How does the RSA private key passphrase work under the hood*. URL: <http://stackoverflow.com/questions/1774469/how-does-the-rsa-private-key-passphrase-work-under-the-hood>
- [27] Symantec. *What is SSL, TLS and HTTPS?*. URL: <https://www.symantec.com/page.jsp?id=ssl-information-center>
- [28] Telegraph. *10 of the best hidden Snapchat features*. Oct 12, 2016. URL: <http://www.telegraph.co.uk/technology/0/the-best-hidden-snapchat-features/>