An Analysis of Strengths and Weaknesses of TLS Utilization in iOS Applications

Brian Krupp
Computer Science Department
Baldwin Wallace University
Berea, OH USA
bkrupp@bw.edu

Malik Matthews
Computer Science Department
Baldwin Wallace University
Berea, OH USA
mmatthew17@bw.edu

Josh Hadden
Computer Science Department
Baldwin Wallace University
Berea, OH USA
jhadden18@bw.edu

***Abstract*—Mobile applications are granted access to large amounts of data both stored and sensed on a user's device. While the data stored on these devices can be protected with encryption, where the private keys are protected through a combination of biometric authentication and passcodes, it is important to consider the security of data in transit and the many connections applications make to different domains. Often, these domains are unknown to the user where they can include third-party domains and tracking services. In this paper, we investigate the current state of transport layer security (TLS) utilization across the top information sharing applications in the iOS App Store. Through this study, we are able to better understand which domains follow best practices and use recommended ciphers and TLS versions. We also provide deeper insight into the prominent use of tracking services and how they differ in their utilization of TLS. Our study utilizes a proxy service to detect the domains that applications communicate with and then uses OpenSSL to analyze the supported ciphers, TLS versions, and certificate details. We also leverage domain analysis from DuckDuckGo Tracker Radar to differentiate tracking domains. From this study, we analyzed 965 unique domains and found that 935 (96%) use a cipher that is not recommended and only 23 (2%) of the domains strictly followed recommendations on cipher and TLS utilization. While most domains are utilizing recommended versions of TLS and recommended ciphers, this study shows there are many that still support vulnerable versions of TLS and vulnerable ciphers, leaving the user's data at risk.**

***Index Terms*—mobile, TLS, SSL, certificates, security, encryption**

I. Introduction

As the world wide web became mainstream, the need to protect data in transit between clients and servers became a necessary requirement. In early implementations, the main protocol for providing secure communications was Secure Sockets Layer (SSL). SSL provided a mechanism for protecting data in transit by ensuring data was secure between a user's web browser and the web server. However, early implementations of SSL were later discovered to contain significant vulnerabilities. While SSL 1.0 was never published, SSL 2.0 was standardized in 1995 and contained several vulnerabilities that were discovered later: message authentication used a weak hashing algorithm (MD5), weak cipher suites exposed the connection to man-in-the-middle attacks, and message integrity and encryption used the same key which could have used a weak algorithm [1]. SSL 3.0 was introduced a year later in 1996, however it also has been deprecated due to vulnerabilities where due to padding used in the cipher block chaining (CBC), SSL 3.0 is vulnerable to POODLE attacks [2]. Additionally, the key exchange is also vulnerable to man-in-the-middle attacks [3].

To address these vulnerabilities in SSL, Transport Layer Security (TLS) was introduced. Starting in 1999, TLS 1.0 was defined [4] which specifically addressed how to prevent man-in-the-middle attacks. However, TLS 1.0 also was discovered to contain vulnerabilities and in 2006 was replaced with TLS 1.1 which addressed issues in how initialization vectors (IV) were implemented and padding errors handled to protect against CBC attacks and other various attacks [5]. TLS 1.2 included further improvements such as adding strong cipher suites and deprecating weak cipher suites [6]. Finally, TLS 1.3 improved upon 1.2 by prohibiting negotiation to insecure versions and ciphers, dropping support for insecure features, adding additional secure signature algorithms, and encrypting all handshake messages [7].

At the time of this writing, the National Institute of Standards and Technology (NIST) provides guidelines that recommend at least TLS 1.2 is utilized [8] as versions of TLS from 1.1 and below are vulnerable to attacks such as POODLE [2] and BEAST [9]. Even with this recommendation, TLS 1.2 is recommended with only using specific ciphers [10], [11].

In this study, we are interested in how the domains that mobile applications communicate with follow these recommendations. With users spending on average over 5 hours a day on their screens [12], their data may be at risk with services not following the latest recommendations. We are also interested in this study the differences in domains that are known trackers and how they differ in their adoption of TLS recommendations to those domains that are not trackers. Given that the online advertising industry is very profitable [13], there is a financial incentive to not adopt the stricter TLS recommendations to support older clients so that both data collection and advertising can be increased. In this paper, we make the following contributions:

- We perform an analysis of 965 unique domains that are utilized by 124 of the top information sharing applications in iOS.
- We analyze each domain to understand what versions and ciphers of TLS they support to understand how strictly

978-1-6654-2854-5/21/$31.00 ©2021 IEEE

they follow recommendations.
- We detect various personal data such as the location, contacts, and photos from the device to better understand how domains that collect this data follow recommendations.
- We compare these results with domains that are known trackers using DuckDuckGo's Tracker Radar to understand what differences exist between tracking and non-tracking domains.

From this study, we find that a vast majority of domains do not follow the current recommendations for implementing TLS. They either use an insecure version of TLS that is susceptible to well-known attacks or a weak cipher that is no longer recommended. Additionally, we find that certificates commonly exceed the maximum recommendation time of 366 days for a certificate to be valid. Finally, we found that a majority of domains that are trackers that are frequently used by mobile applications also do not follow the most recently recommendations of TLS where in our study we found these trackers to receive personal data such as the user's location.

The paper is organized as follows: Section II describes the system design we utilized to perform the study. Section III provides the methods we used to conduct the experiment. Section IV describes the findings from the analysis we performed. Section V describes related work. Section VI discusses potential limitations in the study and items to consider in future studies. Finally, Section VII summarizes our conclusions of the study.

II. System Design

To capture the domains that are utilized by iOS applications, we utilize a web proxy. The proxy is designed to not only capture HTTP requests, but it also detects personal data that is transmitted from the application to a domain [14]. To intercept traffic that utilizes TLS, a unique certificate authority (CA) is generated which is installed on the iOS device as a root certificate. This allows a man-in-the-middle attack to be performed where the proxy dynamically generates certificates based on the domain the mobile application is attempting to connect to and sign certificates for each domain using the trusted root certificate. Frequently generated certificates are stored in a cache to improve performance. By intercepting traffic over TLS, this also allows the proxy service to inspect traffic to discover personal data that may be sent to the domain. As the mobile application is tested, the proxy service logs each domain to an activity log with details such as whether or not TLS is being utilized and if personal data was discovered in the request.

Once domain information is captured, an automated process using OpenSSL examines support for TLS and checks each domain for support of both recommended and not recommended ciphers. Additionally, certificate details are collected including period of validity and key length. This analysis also uses domain information from the DuckDuckGo Tracker Radar project [15] to determine if a domain is a known tracker and what tracking categories it exists in. This overall process is depicted in Figure 1.

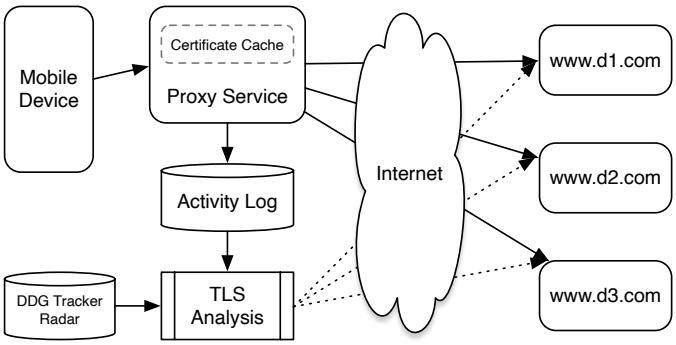

Fig. 1: System design to perform TLS analysis

III. Experiment Design

A. Application Testing

This study focused on the top mobile applications in the iOS App Store that can be classified from information sharing categories. This criteria targeted applications in categories where user data is more likely to be transferred from the device to a particular domain. From this criteria, the top ten free applications were selected from the below categories:

- Business
- Education
- Kids
- Lifestyle
- Magazine and Newspapers
- News
- Reference
- Social
- Sports
- Weather

Automated analysis of applications without requiring a jailbreak of the device is limited in iOS as it is a closed source operating system. Additionally, as part of the study, our goal was to understand how applications behave on unmodified operating systems as a normal user would use the application. Therefore, testing of each application was performed manually where the tester followed strict steps outlined below:

1) Disable HTTP proxy to download applications to be tested.
2) Download each application to be tested.
3) Disable Background App Refresh.
4) Enable proxy in Wi-Fi settings.
5) Open application. Wait 2 minutes before interacting with main UI. Immediately accept any permission requests. During this period, verify data is being captured.
6) If application requires an account to access application, create account using provided test account.
7) Continue to test application for 5 minutes. Navigate main components: if it is a tab bar, access each component in tab, if it is a hamburger type navigation, access each component.
8) Open utility application and tap on "Complete Testing" button.

If an application existed in multiple categories, it was skipped if it was already tested. Applications that were non-functional were still included in the study as the TLS connection that was attempted was still captured. An application was determined to be non-functional if messages displaying network connection errors were shown in the application or screens would not load during testing. Applications that utilize certificate pinning [16] were not able to be tested using the proxy service as trust could not be established if the pinned certificate within the application was not presented. Still, the domain could be captured when trust was attempted to be established. From this criteria, we successfully gathered network communication of 124 iOS applications where we are able to analyze 965 unique domains.

B. Analyzing TLS, Ciphers, and Certificates

To analyze a domain's support for different versions of TLS and ciphers, we utilized OpenSSL and a script to automate the analysis. We first examined the support of TLS versions 1.0 to 1.3 without specifying the cipher to discover what versions of TLS each domain supports. Below is an example of testing the domain ad.doubleclick.net using TLS 1.2:

```
openssl s_client -connect ad.doubleclick.
net:443 -tls1_2
```

Once we were able to determine each version of TLS that a domain supports, we then examined each cipher. Below is an example of testing the the same domain with TLS 1.2 using the ECDHE-ECDSA-AES256-GCM-SHA384 cipher:

```
openssl s_client -connect ad.doubleclick.
net:443 -tls1_2 -cipher ECDHE-ECDSA-AES256
-GCM-SHA384
```

We did not examine cipher support for TLS 1.0 or TLS 1.1 as both versions are no longer recommended. We did however examine cipher support for TLS 1.2 as only specific ciphers are recommended. The ciphers we examined for support were the following based on recommendations from both Mozilla [10] and the Open Web Application Security Project [11]:

- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-CHACHA20-POLY1305
- ECDHE-RSA-CHACHA20-POLY1305
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256

We also examined support for ciphers that are not recommended which are packaged with OpenSSL and included 21 ciphers. This allowed us to examine if domains also supported not recommended ciphers such as AES128-SHA256. To examine certificates for each domain, we also used OpenSSL to extract the X.509 certificate and store it in a text file to examine:

TABLE I: Domains supporting TLS versions from 1.0 to 1.3

TLS Version	# of Domains
TLS 1.0	701
TLS 1.1	720
TLS 1.2	965
TLS 1.3	369

```
openssl s_client -servername ad.
doubleclick.net:443 -connect ad.
doubleclick.net:443 2>/dev/null | openssl
x509 -text > cert.txt
```

From this file, we are able to extract the intermediate and root certificate authority, certificate validity, key sizes, and signature algorithms.

C. Analyzing Tracking Domains

To determine if a domain is classified as a tracker, we utilize the DuckDuckGo Tracker Radar dataset. To utilize this dataset, we mapped each of the 965 unique domains using the `domain_map.json` file that was provided. From this, we are able to extract information for each domain by removing subdomains and using the domain's name as the filename. For example, with the domain ad.doubleclick.net, domain information would exist in `doubleclick.net.json`. From this, we are able to extract if the domain is a known tracker and what tracking categories it exists in. For the purposes of this study, if a domain existed in any tracking category such as *Category Advertising* or *Category Analytics*, it was determined to be a domain that performs tracking.

IV. Results

From the 965 unique domains that we gathered in this study, we breakdown our findings here by analyzing support of different versions of TLS, which ciphers are utilized and the domains that follow recommendations, how do trackers differ in their use in ciphers, and key characteristics of certificates.

A. TLS and Cipher Analysis

Analyzing the support of different versions of TLS, all of the 965 domains that were analyzed support TLS 1.2. Of these, 720 supported TLS 1.1 and 701 supported TLS 1.0. It is important to note that TLS 1.0 and 1.1 are no longer recommended. The latest version of TLS 1.3 was supported by 369 of the domains. These results are shown in Table I.

In this study, we examined which domains supported recommended ciphers to be used with TLS 1.2 [10]. Of the recommended ciphers ECDHE-RSA-AES128-GCM-SHA256 and ECDHE-RSA-AES256-GCM-SHA384 were the most popular with 938 and 927 domains utilizing these ciphers respectively. A breakdown of the number of domains that utilized recommended ciphers is shown in Table II. We were also interested in the ciphers that were supported in TLS 1.2 but not recommended for use. Table III shows a summary of ciphers that are no longer recommended but were supported.

TABLE II: Recommended Ciphers Supported by Domains

Cipher	Domain Count
ECDHE-RSA-AES128-GCM-SHA256	938
ECDHE-RSA-AES256-GCM-SHA384	927
ECDHE-RSA-CHACHA20-POLY1305	535
ECDHE-ECDSA-AES128-GCM-SHA256	350
ECDHE-ECDSA-AES256-GCM-SHA384	349
ECDHE-ECDSA-CHACHA20-POLY1305	326
DHE-RSA-AES128-GCM-SHA256	126
DHE-RSA-AES256-GCM-SHA384	125

TABLE III: Not Recommended Ciphers by Domains

Cipher	Domain Count
AES128-GCM-SHA256	883
AES256-GCM-SHA384	763
ECDHE-RSA-AES128-SHA256	687
ECDHE-RSA-AES256-SHA384	683
AES128-SHA256	553
AES256-SHA256	536
ECDHE-ECDSA-AES256-SHA384	193
ECDHE-ECDSA-AES128-SHA256	193
DHE-RSA-AES128-SHA256	107
DHE-RSA-AES256-SHA256	106
DHE-RSA-CHACHA20-POLY1305	47

B. Recommendation Analysis

In examining the domains that follow recommendations of both TLS support and cipher support, we found that 935 domains use a cipher that is not recommended and 961 use a recommended cipher. This leaves 4 domains that utilize TLS 1.2 but do not support any of the recommended ciphers. In our analysis, only 23 domains followed the recommendations strictly, meaning, they supported TLS 1.3 or used a recommended cipher with TLS 1.2, and did not support ciphers that were no longer recommended. These domains also did not support TLS 1.0 or TLS 1.1.

C. Tracking Analysis

Of the 965 domains that were analyzed, 196 of the domains are classified as trackers using the dataset from DuckDuckGo's Tracker Radar [15]. Of these, 195 of the domains supported TLS 1.3 or TLS 1.2 with recommended ciphers, while 187 supported ciphers that were no longer recommended or TLS 1.0 or TLS 1.1. A breakdown of domains within tracking categories that followed or did not follow recommendations is provided in Table IV. Motivated Tracking, Advertising, and Analytics were the most common categories. There are additional categories such as Malware and High Risk Behavior that we did not discover any domains existing within, however, it is important to note that we did find domains that existed within the Ad Fraud category. We additionally examined which domains were trackers where the proxy detected personal data within the request. The personal data that was detected was either the user's location, contacts data, or a photo. An example of location discovered in a HTTP POST request is included below where a precise location along with other identifiers is sent from the *NOAA Weather Radar* iOS application to the domain `api2.amplitude.com`, which is a known tracker:

TABLE IV: Trackers Following Recommendations

Tracking Category	Recommended	Not Recommended
Motivated Tracking	146	142
Advertising	136	132
Analytics	108	100
Audience Measurement	75	72
Third Party Analytics Marketing	68	63
Social Network	31	30
Action Pix	29	28
Ad Fraud	27	25
Session Replay	8	4


```
api_properties":{"location":{"lat
":41.374237060546875,"lng
":-81.850124411094953},"ios_idfv":"7
F83E3CB-5901-437E-9420-0996A0ED56E9"},"
device_id":"7F83E3CB-5901-437E-9420-0996
A0ED56E9","event_properties":{"Search
Symbols Entered":"4","Time Elapsed":"0"},"
uuid":"28664B9B-A15D-4EB1-9E33-
D665734B2EA9","device_manufacturer":"Apple
","version_name":"4.10.2
```


From this analysis, we found 83 domains that are trackers that received personal data and 82 of these followed recommendations in supporting TLS 1.3 or 1.2 with recommended ciphers. However, 81 of the 82 domains used a not recommended version of TLS or a not recommended cipher.

D. Certificate Analysis

In analyzing certificates, we were interested in the public key algorithm, public key size, and the the validity of certificates. From our analysis, a majority of domains used RSA with 651 domains total, and 314 domains used Elliptical Curve as their public key algorithm. The most common key size for those that used RSA was a 2048 bit key with 648 domains, and 3 domains using a 4096 bit key. For those that used Elliptical Curve, all 314 domains used a 256 bit key size.

In examining certificate validity, the most common certificate length was 365 days with 193 certificates having this length. However, this only accounts for 20% of the domains. 481 of the domains had certificates expire over a year, which does not follow the recommendation of a certificate being valid for at most 366 days and now recommended for 90 days where 220 domains followed the newer recommendation [10]. Figure 2 shows the distribution of certificates by the number of days they are valid.

V. Related Work

To our knowledge, a study that specifically targeted studying the use of TLS in the iOS ecosystem does not exist. However, there has been research targeting Android and the overall use of TLS and certificates. The most directly related work that studied usage of TLS on Android focused on what APIs applications used to incorporate TLS and found that most use the default libraries [17]. The study also investigated how applications verify certificates and the use of certificate pinning where they found high risk applications were less likely to

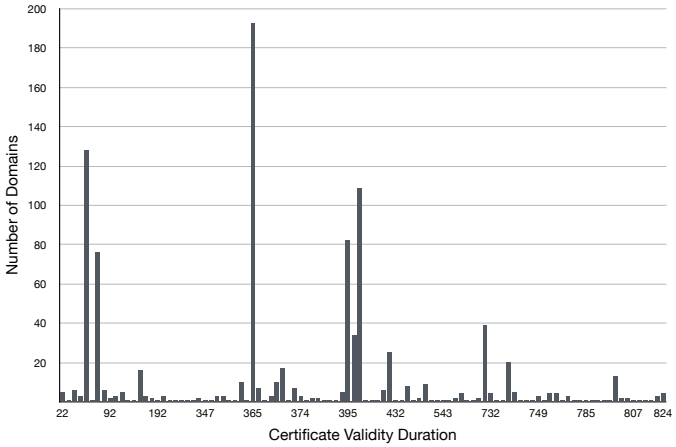

Fig. 2: Distribution of certificate validity length by days.

deploy additional verification. Another study that focused on Android apps discovered that significant differences existed in implementation both with developers in creating mobile applications and server configuration for utilizing TLS [18].

Along with implementing TLS in Android, one study investigated how the trusted root certificate store in Android can be exploited where TLS interception can be performed [19]. Their study found that certificate stores are bloated where devices are more susceptible to attacks. Similar research also discovered the ability to steal credentials based on how mobile applications trust certificates that are presented to them [20]. Other research focused on studying the implementation of certification validation in how they are used in Android applications with third party libraries. They found that popular mobile applications such as banking are susceptible to man-in-the middle attacks where developers that use standardized libraries often perform certificate validation incorrectly [21]. Similar research focused on studying Android applications that were susceptible to man-in-the-middle attacks and how users perceive certificate warnings and security indicators [22]. In this study, they were able to steal credentials from popular payment and credit card applications.

Outside of mobile, research has also focused on analyzing all certificates across the entire ecosystem where attributes such as validity, key size, and issuers were analyzed [23], [24]. In this research, it was discovered that a large number of entities could sign certificates for any website. Along with analyzing certificates, keys have also been the focus in how they are used to establish trust and what steps vendors took once vulnerabilities such as Heartbleed were discovered [25]–[27]. From the study, many vendors never patched their servers even though the keys they used could have been at risk due to Heartbleed. Further analysis looked at Heartbleed where they found that over 73% of certificates were never reissued and 87% were not revoked [28]. Additionally, the studies found that certificate revocation is ineffective where mobile browsers would not check if certificates were revoked. Research has also looked at forged certificates and while they can be used for antivirus software and corporate filters, they also found they can be utilized in popular websites to intercept traffic and for malware [29].

VI. Discussion

Given that iOS is a close-sourced operating system, automating the testing of applications without jailbreaking the device is limited. In this study, we were able to gather a large enough sample (n=124) of applications in information sharing categories that provides a representation of the current state of the iOS app store. This sample allowed us to study 965 unique domains with commonly used trackers. While applications may be categorized in an incorrect category, a vast majority of the applications that were tested did represent the categories they existed within.

Current and past versions of SSL/TLS have shown that in time, vulnerabilities will be found and current recommendations and implementations will be exploited. With older versions of TLS having documented vulnerabilities that eventually get patched, there are systematic flaws that have not been fixed. These flaws stem from how TLS 1.2 and previous versions were more reactive to vulnerabilities that were later discovered. This process leaves a window of opportunity for malicious actors to exploit the known vulnerability until a newer version or recommendation is provided. With this in mind, TLS 1.3 took a different approach where the community was tasked with identifying issues before deployment. This approach is why TLS 1.3 is considered to be more proactive compared to its past versions.

Within this study, there were applications that could not be fully tested due to the use of certificate pinning. However, we were still able to capture the domains that applications attempted to communicate with to better understand their TLS implementation. Still, if a network request failed, this could prevent other areas of the application to be tested and potentially additional domains to be analyzed.

Studying the level of security that applications exert when communicating with a server is important to users. The more transparency a user is given regarding the handling of their data, the more aware each individual can be to the safety of their personal data. Additionally, if users are able to see how applications and the domains they communicate with follow recommended practices, it is likely that these applications would make more secure and ethical decisions when it comes to handling security and privacy.

VII. Conclusions and Future Work

In this paper, we analyzed 965 unique domains that are used by the top information sharing applications in the iOS App Store. From this analysis, we found that while all domains support TLS 1.2, 883 domains supported at least one cipher that is no longer recommended, putting user data at risk. Additionally, a majority of the domains still supported older versions of TLS where 720 supported TLS 1.1 or below. These versions of TLS contain well known vulnerabilities and are susceptible to attacks such as BEAST and POODLE.

In analyzing domains that follow strict recommendations, only 23 domains used TLS 1.2 with only recommended ciphers or TLS 1.3. Additionally, we found that 481 of the domains had certificates that expire over 366 days which is the maximum recommended duration. This shows widespread existence of domains that have known vulnerabilities in their TLS implementations, putting user data at risk. Even though iOS has strict requirements for TLS, other clients that connect to these domains where user data is transferred leaves the user's data vulnerable to well-known exploits. Additionally, using the data from DuckDuckGo Tracker Radar, we found that 196 of the domains are trackers and all but one domain do not follow recommended practices in implementing TLS. Of these trackers, 83 received personal data such as the user's location.

This study provides important insight into the current state of TLS utilization by applications in the iOS App Store. Since there is no accessible method to discover the strength of TLS implementation in mobile applications, our study provides that insight and demonstrates that a vast majority of domains that mobile applications communicate with do not follow the current recommendations for implementing TLS. As more transparency is provided to users on the use of personal data and tracking, future work can focus on providing more transparency with the security of network communications. Just as with new features such as *Privacy Labels* and *Privacy Report* in iOS, users can become become more aware of issues around security. We believe when users have more transparency, they can then make informed choices and demand from mobile OS producers or application developers increased privacy controls or more secure practices.

REFERENCES

[1] I. E. T. F. (IETF). Request for comments: 6176. [Online]. Available: https://tools.ietf.org/html/rfc6176

[2] CISA. Ssl 3.0 protocol vulnerability and poodle attack. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/TA14-290A

[3] I. E. T. F. (IETF). Request for comments: 7568. [Online]. Available: https://tools.ietf.org/html/rfc7568

[4] ——. Request for comments: 2246. [Online]. Available: https://tools.ietf.org/html/rfc2246

[5] ——. Request for comments: 4346. [Online]. Available: https://tools.ietf.org/html/rfc4346

[6] ——. Request for comments: 5246. [Online]. Available: https://tools.ietf.org/html/rfc5246

[7] ——. Request for comments: 8446. [Online]. Available: https://tools.ietf.org/html/rfc8446

[8] W. Polk, K. McKay, and S. Chokhani, "Guidelines for the selection, configuration, and use of transport layer security (tls) implementations," 2014-04-28 2014.

[9] T. A. Nidecki. What is the beast attack. [Online]. Available: https://www.acunetix.com/blog/web-security-zone/what-is-beast-attack/

[10] Mozilla. Security server side tls. [Online]. Available: https://wiki.mozilla.org/Security/Server_Side_TLS

[11] O. W. A. S. Project. Transport layer protection cheat sheet. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

[12] "How much time do americans spend on their phones in 2020?" Apr. 2020. [Online]. Available: https://techjury.net/blog/how-much-time-does-the-average-american-spend-on-their-phone/

[13] iab. Internet advertising revenue report. [Online]. Available: https://www.iab.com/wp-content/uploads/2020/05/FY19-IAB-Internet-Ad-Revenue-Report_Final.pdf

[14] B. Krupp, D. Jesensky, and A. Szampias, "Speproxy: Enforcing fine grained security and privacy controls on unmodified mobile devices," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, Oct 2017, pp. 520–526.

[15] DuckDuckGo, "Duckduckgo tracker radar," February 2021. [Online]. Available: https://github.com/duckduckgo/tracker-radar

[16] "Certificate and Public Key Pinning | OWASP." [Online]. Available: https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning

[17] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying tls usage in android apps," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 350–362. [Online]. Available: https://doi.org/10.1145/3143361.3143400

[18] X. Wei and M. Wolf, "A survey on https implementation by android apps: Issues and countermeasures," *Applied Computing and Informatics*, vol. 13, no. 2, pp. 101–117, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210832716300722

[19] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson, "A tangled mass: The android root certificate stores," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 141–148. [Online]. Available: https://doi.org/10.1145/2674005.2675015

[20] J. Hubbard, K. Weimer, and Y. Chen, "A study of ssl proxy attacks on android and ios mobile applications," in *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 2014, pp. 86–91.

[21] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: Validating ssl certificates in non-browser software," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 38–49. [Online]. Available: https://doi.org/10.1145/2382196.2382204

[22] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith, "Why eve and mallory love android: An analysis of android ssl (in)security," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 50–61. [Online]. Available: https://doi.org/10.1145/2382196.2382205

[23] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the https certificate ecosystem," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 291–304. [Online]. Available: https://doi.org/10.1145/2504730.2504755

[24] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman, "Towards a complete view of the certificate ecosystem," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 543–549. [Online]. Available: https://doi.org/10.1145/2987443.2987462

[25] M. Hastings, J. Fried, and N. Heninger, "Weak keys remain widespread in network devices," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 49–63. [Online]. Available: https://doi.org/10.1145/2987443.2987486

[26] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's pki," 10 2015, pp. 183–196.

[27] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your ps and qs: Detection of widespread weak keys in network devices," in *Proceedings of the 21st USENIX Conference on Security Symposium*, ser. Security'12. USA: USENIX Association, 2012, p. 35.

[28] L. Zhang, D. Choffnes, T. Dumitraş, D. Levin, A. Mislove, A. Schulman, and C. Wilson, "Analysis of ssl certificate reissues and revocations in the wake of heartbleed," *Commun. ACM*, vol. 61, no. 3, pp. 109–116, Feb. 2018. [Online]. Available: http://doi.acm.org/10.1145/3176244

[29] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged ssl certificates in the wild," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 83–97.