An Empirical Study of Mobile Apps SSL Stripping Man-in-the-middle Attack

[1]Muhammad Salisu Ali, [2]Sani Salisu, [3]Uba Yusuf Magaji, [4]Ayuba John

[1,4] Department of Cyber Security
Federal University Dutse

[2] Department of Information Technology
Federal University Dutse

[3] Department of Software Engineering
Federal University Dutse

Corresponding Author: ms.ali@fud.edu.ng

Abstract
The increased pervasive and powerful nature of smartphones, palmtops, tablets and, personal digital assistance (PDAs), have made handheld devices necessities in management for enterprises, education sectors, communications and, businesses. These devices allow the use of third-party applications. Third-party apps enhance the mobile user experience for instance; mobile apps can scan QR Codes and provide coupons. Moreover, users can stay in touch with email apps and maintain their calendars and to-do lists, play games, book flights, book medical appointments and lots more on the move. Several thousands of these mobile devices and third-party apps rely on Secured Socket Layer (SSL) to protect sensitive data from eavesdropping and tampering. Recently though, a new kind of attack against the way SSL is used evolved – SSL Strip, which can be used to bypass all the encryption provided by the SSL. The attack is powerful in that it produces no certificate warning and negative feedback to the client something that people used as a security indication. People use these devices and apps to perform sensitive transactions on public wireless networks without suspecting that the communication could be eavesdropped on by an adversary. In this research work, we presented an experiment to find if mobile apps third party applications from Android and iOS are vulnerable to this attack. We have collected 3 among the most popular benign apps for each of the mobile platforms.

Keywords: SSL, SSL Strip, Man-in-the-middle attack, android, iPhone, mobile devices, third party apps.

INTRODUCTION

Mobile devices are progressively becoming an indispensable aspect of human life due to their rapid increase in rich and versatile functionality. Mobile devices are the most essential way of communication, source of information and, important storage tools (Ali

Balapour, et. Al, 2020). Innovation of new apps for these mobile devices facilitate users to perform routine work such as mobile banking, payment of utility bills, keeping health records, games and, lots more. The powerful nature of mobile apps has made it easy for consumers to use internet-related services just like they are using them on their traditional desktop computers.
Mobile devices are constrained by a small screen, weight, a data input mechanism, processing power, memory space, battery capacity and, user interfaces. As a result the mobile operating system and browsers lax secure applications and indicators for identifying malicious and illegitimate apps. User cannot tell what mobile app or website they are interacting with. This exposes users to the risk of mistaking a malicious/fraudulent app for a legitimate one, Xuetao et. al (2017). Mobile apps may have access to the user's data and are often able to access the internet, send SMS or even make calls.

If the media through which mobile apps sends and receive data is not secured (encrypted), there is potentially a possibility of man-in-the-middle attacks (MiTM). Generally, mobile devices lack wired network capability, and therefore must be connected to a wireless network using built-in hardware in. Wireless is a shared medium which means everything that is sent or received could be intercepted by a malicious person. In a study by George H. et el. (2011), the world is bound to see an increase in man-in-the-middle attacks due to the widespread hacking technology and sophisticated tools.

This research work is limited to third party applications (for Android and iOS platforms). Many mobile applications normally depend on Wi-Fi to access internet resources. Using mobile apps to utilize these resources could potentially lead to man-in-the-middle attack (MiTM). Although mobile apps may use HTTPS to provide end-to-end data protection, an attacker could use a kind of attack – SSL-Strip to strip the SSL connection between the client and server.

Study of Mobile Apps Security
Mobile Operating Systems naturally become the target for security scrutiny as they are responsible for managing all the information and services. According to Ken mar M. A. et. Al. (2017), security professionals and hackers are both working very hard to exploit mobile devices. Apple iOS and Android are the two most popular mobile operating systems. The following sections explain the security of each of the mobile OS.

iOS Security
This is a mobile OS developed by Apple originally for iPhone and later extended to support the iPod touch, iPad tablet and second-generation Apple TV. The iOS kernel is based on Darwin OS and was developed in Objective-C. Apple iOS has four abstraction layers: the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer (Miss Priyanka, et. al. 2013).

Buffer overflow, integer overflows, and format string attacks occurred typically in software and applications that are developed in C/C++ programming languages. These programming languages do not provide for an array

boundary checking and therefore leave security totally at the discretion of the programmer. Jiadong Ren, et, al (2019) described buffer overflow as the most dangerous software vulnerability. If it can be effectively overcome, The Threat to software development can be greatly alleviated. Despite all efforts from Apple to make sure that iOS is essentially secured by providing some built-in protective measures, exploits resulting in code execution are still possible (Liu, et. al, 2016).

Apple iOS third party apps use a Cocoa Touch Application Programming Interface (API) to interact with its iOS devices. Written in Objective-C, the frame provides a means of abstraction from the mobile operating system (iOS). However, several security core features have been implemented in the iOS platform to provide protections against attacks. These security features includes: code signing, sand boxing, address space layout randomisation (ASLR), and encryption (Yixiang and Kang, 2017).

Android Security

The android mobile OS is a Linux-based traditional desktop modern operating system in accordance with the Open Handset Alliance (OHA), designed for mobile devices. The Android kernel includes a camera driver, which allows the user to send commands to the camera hardware. Android and its apps are typically programmed with the Java or Kotlin programming languages (Omar and Amira, 2017).

Unlike Apple, Android do not offer code signing and certification rule for apps developer, and this could be the reason behind the widespread of Android malware. Google allow anyone to create and release apps anonymously without monitoring and they could be downloaded from many different sites apart from the official Android market (Chris Rose, 2012). Android follows a crowdsourcing approach instead of the top-down approach of malware prevention to protect the App-Store from malware. Developers can develop and distribute applications on the Google market and elsewhere without any review. Zafar Kazmi et al. (2011) mentioned that the App submission process by Android is not sufficient to identify and preclude the submission of malicious applications to the Android Market.

Sascha Fahl et al. (2012) analysed 13,500 popular android free apps and found that 1,074 (8.0%) of the apps that were tested contain SSL/TLS code that is potentially susceptible to MiTM attacks. In another hand, William Enck et al (2015). tested 1,100 Android third party apps and detected widespread use of privacy-related information such as IMEI, IMSI, and ICC-ID for cookie tracking. However, no exploitable vulnerabilities that could have resulted in vicious control of the device were observed.

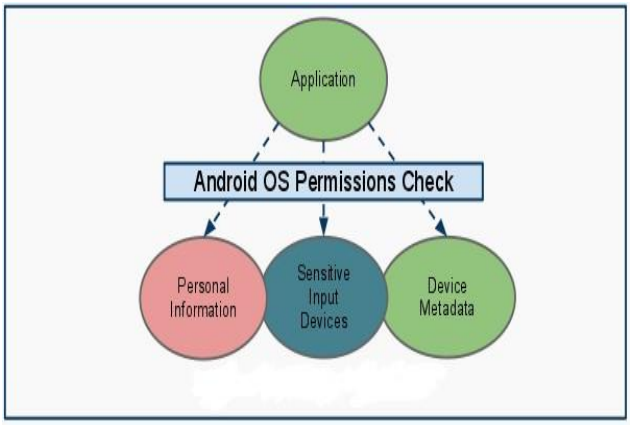

Figure 1: Access to sensitive user data is only available through protected APIs (taken from Google images)

Security issues with mobile apps

The fastest-growing of mobile devices equipped with powerful features and third party apps increases the rate at which these devices are used by individuals in education, health services, as well as in enterprises (Hurlburt G., et al., 2011). Third party apps grow as much as mobile devices do with millions of apps to download for iPhones and androids on Apple App Store and Google play store or the internet respectively.

The proliferation of web applications and web-related technologies such as cloud computing and bring-your-own-device commonly known as (BYOD) encourages employees, business partners and clients to access information on their devices not managed by the company's IT departments. As a result, give a potential gateway for hackers to attack their device which might result in loss of privacy, security breach, or loss of sensitive data (Morrow, 2012). Once mobile device security is compromised, the uncovered sensitive data can be used to gain access to the owner's accounts. The device can also be used as a gateway into other data resources by taking the advantage of trust laid between the device and the IT infrastructure.

Secured Socket Layer (SSL)

Secure Socket Layer (its successor Transport Layer Security (TLS)) is a cryptographic protocol that provides secure data communication between client host and server. SSL is currently the de facto standard security protocol that is widely deployed for establishing an encrypted link between a web server and browser. It is the security protocol for securing HTTP known as HTTPS symbolised by a little lock that appears around the corner of web browsers during any secured transaction. It was introduced to protect client and server communication from eavesdropping and tampering. However, according to research, many application developers do not use HTTPS or go against the rules which prevent users from man-in-the-middle attacks (Xuetao et. al. 2017).

SSL/TLS Stripping Attack

The SSL Strip attack was introduced by Moxie Marlinspike at the Black Hat 2009 DC conference (Moxie Marlinspike, 2009). SSL strip attack takes advantage of the way users often browse the web pages to get access to secure servers. Most users do not explicitly type the safe address of a web (https) they are browsing, but instead, rely either on the browser or the server to redirect them to a secure page. This opens the opportunity for users' sessions to be stripped whilst giving the user the illusion of privacy (Xuetao et. al. 2017).

The attack exploits the weaknesses in the way browsers validate certificates and warn users about invalid certificates. A low level of awareness about SSL protection by the users is an effective factor that this type of attack can leverage. The attack is particularly crafty because it acts as a MiTM, keeping an eye on HTTPS requests and then mapping them to HTTP look similar setups. They are often redirected to a secured protocol from unsecured web pages or follow HTTP links. In these cases the attacker replaces a secured HTTPS and alters redirection headers to keep user communication unencrypted. MiTM then communicates with the

server securely and the server itself is not able to detect anything wrong. SSLStrip MiTM attack users the simple fact that most users do not type a safe address of the web pages (HTTPS) they are visiting explicitly, but rather depend on the client or target page to redirect them to the secure pages (Nick Nikiforakis, et. al (2010). There is no simple way how to generally prevent this kind of attack (J. Hodges, et. al., 2010).

Attack Scenario

Two (Attacker and victim) machines were set. The first machine consists of the tools for carrying out the attack. The second machine is preloaded with the sample apps. On the attacker's machine, a MiTM attack is used to eavesdrop and forward communication to the server that the victim is connecting to. When the server responds to the client request, the HTTPS connection will be change to a normal HTTP connection using the SSL stripping technique. Below is the set of steps followed to launch the attack:

I. On the attacker's machine, we launch a MiTM attack by setting the machine between the victim and the server. We first make sure the connection is set on the same network and then wait for the victim to request a connection to the server.
II. On the victim's machine, a request would be sent to the server for example amazon.com and wait for a response from the server.
III. When packets from the victim's machine request to a web server on an HTTP connection, on the attacker's machine, we forward the first request to the server and wait for the server to process the request. After processing the request by the server and returning a redirection to an HTTPS connection, on the attacker's side, the response packets from the webserver will be replaced with a normal HTTP connection and then send to the victim.
IV. When everything goes right, a normal HTTP URL will be shown on the web browser or web apps of the victim. However, in that case, any communication that the victim will make will be stripped and use the normal HTTP protocol which lacks an encryption algorithm. Therefore, when the victim sends any sensitive information such as login credentials, chat messages or credit card numbers, the attacker would obtain it in clear text as it was stripped out through the process.
V. When the attacker receives the HTTP packets from the victim's machine, the plain text received will be encrypted to ciphertext on the HTTPS connection and then send to the web server. Although the web browser of the victim uses an HTTP connection, the webserver uses an HTTPS connection. Therefore, no warnings or errors are presented by the browser to the victim, consequently, this kind of attack is difficult to detect. If the attack is successful, the only indicator to the victim that the communication was not over HTTPS is the absence of the lock icon and the "s" signs in the

https://... in the address bar of the browser. The figure below shows an attacker between a mobile device and the webserver:

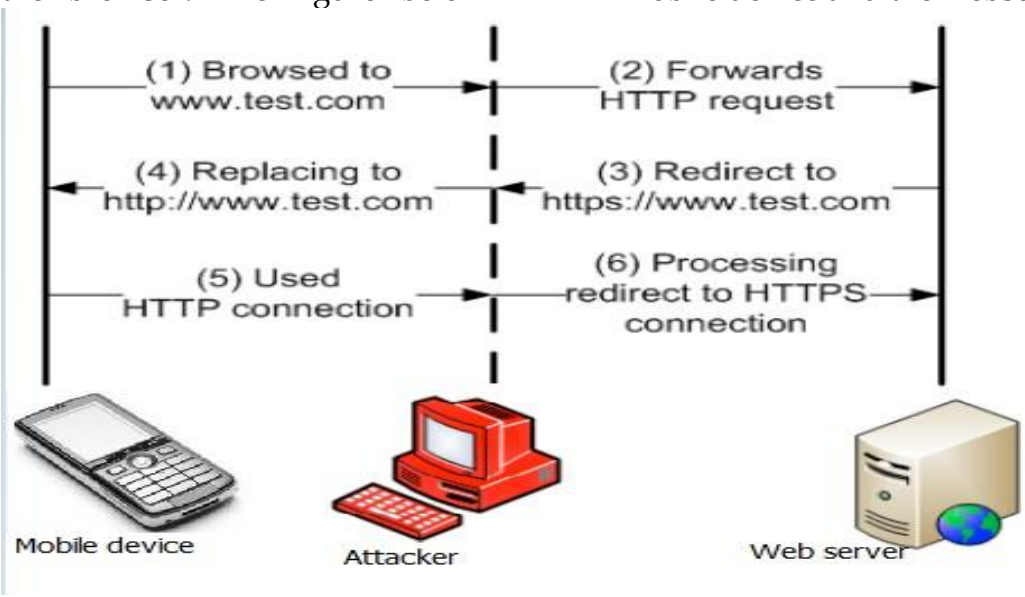

Figure 2: SSL Strip Attack

The figure below depicts how sslstrip converted the https links to http. Login credentials between the browser (mobile apps in this case) and the sslstrip are sent in clear text.

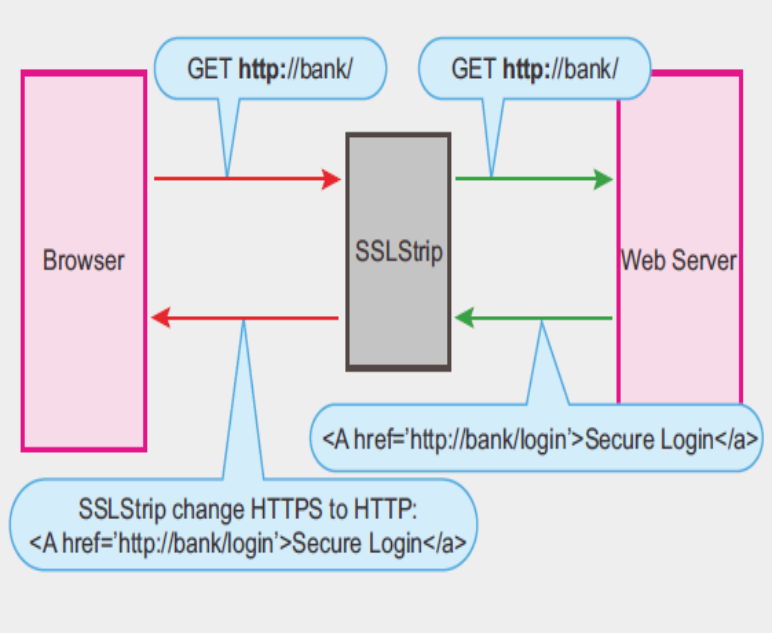

Figure 3: SSLStrip converts HTTPS links to HTTP

Research findings

6 mobile applications each for Android and iOS were downloaded from their respective App-Stores. Of the 6 mobile apps, 1 is an iOS Amazon app, 1 iOS Facebook app and 1 Skype App. For consistency, similar apps for Google Android were tested. These are 1 Amazon app, 1 Facebook app, and 1

Skype app. Apps for both platforms were tested to see if there are differences in what they send to the internet or check if particular app for one platform is weaker than the other.

iOS Apps Test Result

All the 3 iOS app attacks were unsuccessful. Amazon and Facebook apps do not show any logging errors. These apps mostly stated that there were technical or connectivity problems and advised the user to try to reconnect later.

All the data captured for both apps were encrypted. Skype logging was successful but the data captured with tcpdump and ettercap shows the data was encrypted. However, all 3 apps were tested alongside Apple's default browser (Safari) and found they are all vulnerable to the attacks. We were able to capture username and passwords, cookies and session IDs.

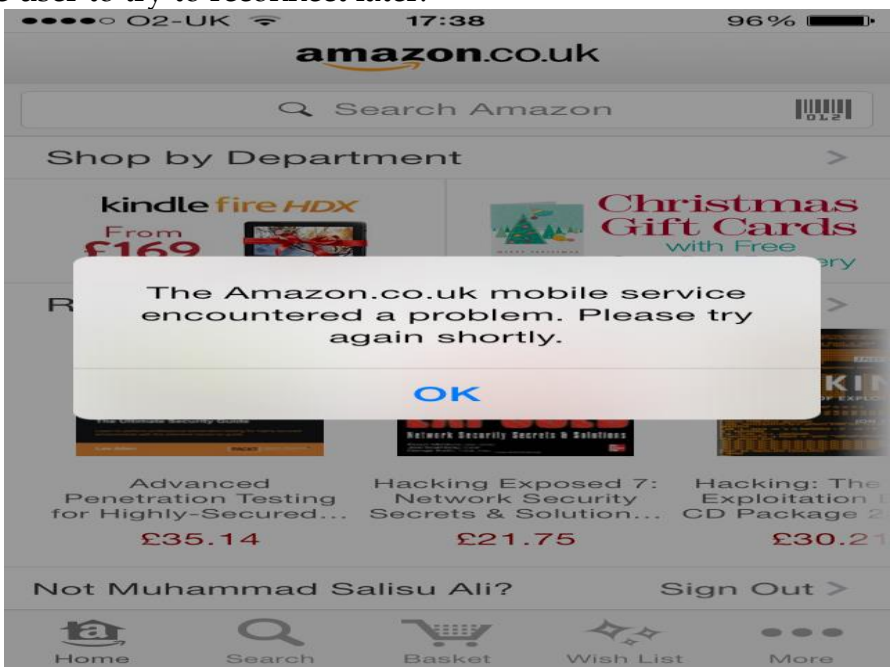

Figure 4: iOS Amazon Error message

Android Apps Test Result

As with the iOS applications, all 3 tests were unsuccessful. The test was also conducted with Android default browser. However, unlike the iOS default browser test, the entire 3 tests show a certificate warning that

> "There are problems with the security certificate for this site. This certificate is not from a trusted authority."

Usernames and passwords are captured in plain text. In all the tests, cookies and session ids are also captured in plain text. Sascha Fahl, et al. (2012), mentioned that the attack is mainly an issue with android browser apps, but can also affect those apps that use Android's webkit. WebView that does not start a browsing session on HTTPS site. The attacks on the Android default browser might be unsuccessful if Android is using a tool HTTPS everywhere as proposed by the HTTP Strict Transport Security IETF Draft (J.

Hodges PayPal, et. al. 2012). According to the research, we found that before 2012 Android did not used such tool on their devices (Sascha Fahl, et al., 2012). On the other hand, according to the same author (Sascha Fahl, et al., 2012), Androids uses sensible trust managers and host name verifiers to notify client of incorrect or fake certificate attacks. But this does not provide protection against MiTM because the client can choose to continue after the notification as illustrated in the figure below.

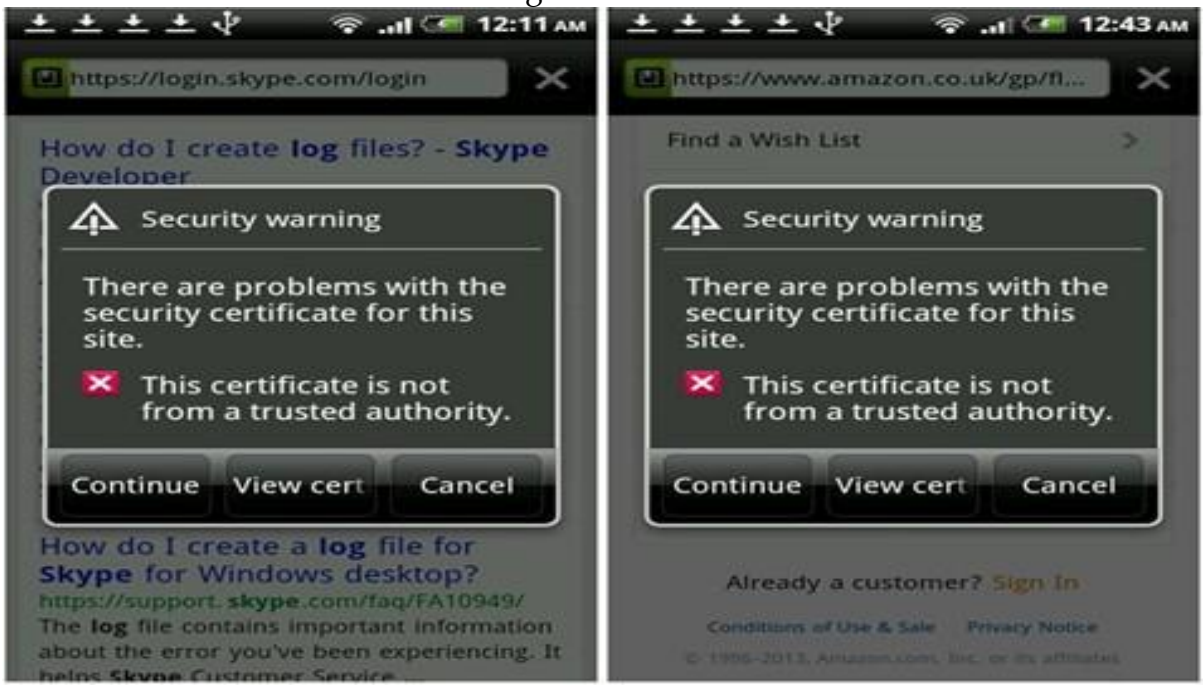

Figure 5: Android certificate warning for Skype and Amazon

CONCLUSION AND RECOMMENDATIONS

Conclusion

In this research work, we presented an experiment to find if mobile apps third party applications from Android and iOS are vulnerable to SSL Strip man in the middle attack. We have collected 6 among the most popular benign apps for Android and iOS mobile platforms. Our analysis of these mobile apps has shown that both Android and Apple iOS has done a great job to countermeasure the threats of SSL Strip MiTM attack. However, to evaluate the real threat of this attack, we repeated the same experiment on both devices (Android and iOS) built-in web browsers. This test has revealed tremendous wealth of sensitive information. We have successfully captured usernames, passwords, sessions and cookies, versions of the SSL/TLS, the mobile OS version, and MAC addresses. We could have gotten credentials for bank, email and PayPal accounts. We found that both devices do not show any security indications that the user is under attack. Android shows a suspicious certificate warning but the user could choose to continue.

Recommendations

We recommend that developers ensure that the apps check for the secure connection or use mutual authentication

before they establish any connection with the server. Deploy HTTP Strict Transport Security (HSTS). The apps should be developed in such a way that the apps accept only trusted certificates and no one can eavesdrop between the client and the server. Apps should also check if the certificate is valid not expired. If expired, the communication should break. End users should avoid using open hotspots and other public wireless points that could be a potential threat to personal data, Users should make sure they check trust certificate warnings and use StripGuard software and HTTPS everywhere.

Future Work

The findings of our research suggest several areas for future work. Gathering a more representative sample data present a challenge we need to address. Many third-party apps that use client-server architecture exist on the Android and Apple App Stores each with their own security strengths and weaknesses. To get more satisfying result, more applications should be used from different categories (finance, education, social networking, lifestyle, kids and many more). Furthermore, several different techniques are there to assess the security of data in transit, this research work is biased on choosing only the SSL stripping attack. Other possibilities such as SSL decrypt attack, SSL Proxy attack and other SSL/TLS attacks can be considered in the future. This research do not provide solution or tool that can be used to thwart the threat of the attack. A tool can be invented in future to countermeasure the attack.

Moreover, the attack is limited to benign apps and non-rooted mobile devices. A jail broken version of the devices and malicious apps can be considered. The results of the experiment can be compared to that of legitimate apps to find out similarities and differences.

REFERENCES

Ali Balapour, Hamid Reza Nikkhah, Rajiv Sabherwal (2020).
Mobile application security: Role of perceived privacy as the predictor of security perceptions. Available online: https://www.sciencedirect.com/science/article/abs/pii/S0268401219309041
[27/03/2022].

Asma Razgallah, Raphael Khoury, Sylvain Halle, Kobra Khan Mohammadi (2020).
A survey of malware detection in Android apps: Recommendations and perspectives for future research. Available online: https://www.sciencedirect.com/science/article/abs/pii/S1574013720304585[27/03/2022].

Chris Rose (2012) Review of Business Information system - first quarter 2012 Smart Phone, Dumb Security, volume 16, number 1. Available online: https://clutejournals.com/index.php/RBIS/article/download/6761/6836 [04/05/2022].

D'Orazio, C. J., & Choo, K.-K. R. (2018). Circumventing iOS security mechanisms for APT forensic investigations: A security taxonomy for cloud apps. Future Generation Computer Systems, 79, 247–261. doi:10.1016/j.future.2016.11.0. Available at: https://www.sciencedirect.com/

science/article/abs/pii/S0167739X16305647 [28/03/2022].

Hurlburt, G., Voas, J., & Miller, K. W. (2011). Mobile-App Addiction: Threat to Security? IT Professional, 13(6), 9–11. doi:10.1109/mitp.2011.104

Available from: https://ieeexplore.ieee.org/abstract/document/6096589 [03/03/2022].

Jiadong Ren, Zhangqi Zheng, Qian Liu, Zhiyao Wei, Huaizhi Yan, (2019). A Buffer Overflow Prediction Approach Based on Software Metrics and Machine Learning, Security and Communication Networks, vol. 2019, Article ID 8391425. https://doi.org/10.1155/2019/8391425 [28/03/2022].

J. Hodges PayPal, et. al. (2012) Strict Transport Security (HSTS).

Internet Engineering Task Force (IETF). Request for Comments: 6797 Category: Standards Track, ISSN: 2070-1721 Carnegie Mellon University. Available online: https://datatracker.ietf.org/doc/html/rfc6797 [13/05/2022]

Liu, F., Liu, K.-S., Chang, C., & Wang, Y. (2016). Research on the Technology of iOS Jailbreak. 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC). doi:10.1109/imccc.2016.178. Availabe from: https://ieeexplore.ieee.org/abstract/document/7774861 [28/03/2022].

Miss Priyanka V. Kanoi, Miss Payal N. ingole(2013). Internal structure of iOS and Building tools for iOS apps. International Journal Of Computer Science And Applications Vol. 6, No.2, Apr 2013 ISSN: 0974-1011 (Open Access). Available from: http://www.researchpublications.org/IJCSA/NCAICN-13/181.pdf [28/03/2022].

Morrow, B. (2012). BYOD security challenges: control and protect your most sensitive data. Network Security, 2012(12), 5–8. doi:10.1016/s1353-4858(12)70111-3. Available from: https://www.sciencedirect.com/science/article/abs/pii/S1353485812701113. [20/03/2022]

Moxie Marlinspike. New Tricks For Defeating SSL/TLS. Presented at the Black Hat DC Conference in Washington, DC on February 18, 2009. Available from: http://blackhat.com/html/bh-dc-09/bh-dc-09-archives.html#Marlinspike [27/0/2022].

Nick Nikiforakis, et. al (2010) HProxy: Client-side detection of SSL stripping attacks.

IBBT-DistriNet Katholieke Universiteit Leuven Celestijnenlaan 200A B3001 Leuven, Belgium. Available online: https://www.securitee.org/files/hproxy_dimva2010.pdf [13/05/2022].

Omar M.Ahmed, Amira B. Sallow (2017). Android Security: A Review

Available from: https://journals.nawroz.edu.krd/index.php/ajnu/article/view/99/116 [27/03/2022]

Patrick Mutchler, Adam Doupe, John Mitchell, Chris Kruegel and Giovanni Vigna (2015), A Large-

Scale Study of Mobile Web App Security. Available from: https://sites.cs.ucsb.edu/~vigna/publications/2015_MoST_MobileWebApps.pdf [27/03/2022].

Sascha Fahl et al. (2012). Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security. CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA. Available online: https://cve.report/CVE-2012-5456/cf2da38f.pdf [04/05/2022].

William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri (2015). A Study of Android Application Security. The Pennsylvania State University. Available online: https://www.cs.utexas.edu/~swarat/pubs/enck-sec11.pdf [04/05/2022].

Xuetao Wei, Michael Wolf (2017). A Survey on HTTPS Implementation by Android Apps: Issues and Countermeasures. Applied Computing and Informatics. Volume 13, Issue 2, July 2017, Pages 101-117. Available online: https://www.sciencedirect.com/science/article/pii/S2210832716300722 [28/03/2022]

Yixiang, Z., Kang, Z. (2017). Review of iOS Malware Analysis. 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC). doi:10.1109/dsc.2017.104. Available from: https://ieeexplore.ieee.org/abstract/document/8005524 [28/03/2022]