2014

Mitigating man-in-the-middle attacks on smartphones – a discussion of SSL pinning and DNSSec

Veelasha Moonsamy
Deakin University

Lynn Batten
Deakin University

DOI: 10.4225/75/57b65a25343ce

12th Australian Information Security Management Conference. Held on the 1-3 December, 2014 at Edith Cowan University, Joondalup Campus, Perth, Western Australia.

MITIGATING MAN-IN-THE-MIDDLE ATTACKS ON SMARTPHONES – A DISCUSSION OF SSL PINNING AND DNSSEC

Veelasha Moonsamy and Lynn Batten
School of IT, Deakin University, Melbourne, Australia
v.moonsamy@research.deakin.edu.au, lynn.batten@deakin.edu.au

Abstract

Since their introduction, smartphones remain one of the most used handheld devices and this trend is predicted to continue in the coming years. Consequently, the number of attacks on smartphones is increasing exponentially; current market research shows that data traffic generated by smartphones will escalate by tenfold in 2019. Such an increase in traffic indicates that the smartphone industry will remain an attractive target for attackers. Whilst smartphone users are aware of the benefits of installing antivirus applications for malware evasion, they have limited knowledge on how to mitigate MiTM attacks. Furthermore, application developers do not always consider implementing appropriate security checks as an important step during the development stage.

In this paper, we describe MiTM attacks based on SSL and DNS and provide a discussion on how they can be mitigated using SSL Pinning and DNSSec. We complete our discussion on mitigation of MiTM attacks by including challenges, limitations and recommendations for application developers and smartphone users. In particular, we suggest that application developers pass a certification test regarding their use of SSL Pinning and/or DNSSec.

Keywords
Smartphone, MiTM, SSL, SSL Pinning, DNS, DNSSec

INTRODUCTION

For the first time since the release of smartphones, global sales are predicted to reach 1 billion units in 2014 (IDC, 2014) and as a result, smartphone data traffic will increase by tenfold in 2019 (Richter, 2013). These forecasts depict the on-going competition amongst the top 4 leading smartphone operating systems (OS) - Android, iOS, Windows Phone, Blackberry- and their corresponding device manufacturers. Nowadays, smartphones with high computational power are sold at affordable prices and hence are easily accessible to interested buyers. Furthermore, smartphone sales have a direct impact on the smartphone application ecosystem; the application markets are driven by the demand and supply of application downloads and uploads, respectively.

Smartphone applications can be considered to be the equivalent of software programs on the desktop platform. They perform a set of tasks depending on the user's needs and require some initial input, such as personal information, to function properly. As such, there is a growing concern around the transmission of user's personal information to and from the device.

In the first half of 2014, several major security flaws were discovered on the smartphone platform. These include Apple's SSL vulnerability (Schwartz, 2014) and the HeartBleed bug in the OpenSSL library (Codenomicon, 2014). In both cases, the implementation of the Secure Sockets Layer (SSL) protocol was compromised and private communication and personal information were divulged through Man-in-The-Middle (MiTM) attacks.

The aforementioned events constitute the primary motivation for this paper. We ask the question: How to re-enforce the SSL protocol to prevent intruders from snooping on traffic to and from smartphones. To answer the question, we investigate the use of two well-known methods which provide authentication between sender and receiver: first, SSL Pinning and second, Domain Name System Security Extensions (DNSSec), which can be regarded as a secured Domain Name System (DNS). The underlying rationale for both of these methods is that by correctly identifying the source and destination of traffic, a third-party will be unable to intercept and capture communications for malicious purposes.

In the rest of the paper, we describe how SSL and DNS work and how MiTM attacks against them can be implemented. We then elaborate on how SSL certificate Pinning and DNS Security Extensions can be used as a preventative measure for these attacks, while also noting their limitations. Finally, we make recommendations

which focus on the role of the application developer in ensuring that security mechanisms are implemented in order to prevent the attacks we have discussed.

MiTM ATTACKS

We provide a brief description of MiTM attacks and elaborate on three types of attacks, based on SSL and DNS, which are commonly exploited on the smartphone platform.

MiTM can be regarded as an active eavesdropping attack. In order to achieve this, the attacker will insert himself between the client/server communication flows. If successful, the attacker will be able to relay traffic to and from the client and server without either endpoint noticing the presence of a third-party. MiTM attacks have been successfully deployed on the desktop platform. However, as the smartphone is rapidly gaining market share, attackers are now shifting their focus to smartphone users as their MiTM victims. For this reason, we describe three popular attacks (SSL Hijacking, SSL Stripping, DNS Spoofing) targeted at smartphone applications.

SSL Hijacking

Traditionally, the SSL protocol is implemented to secure information transmitted between a browser and a web server on the desktop platform. As an initial step, there is an exchange of SSL certificates issued by a Certificate Authority (CA). Browsers come with a pre-installed list of trusted CAs and will communicate with a web server that has been issued a certificate from one of the CAs.

However, on the smartphone platform, applications can be either web- or client-based. A **Web-based** application uses Web 2.0 technologies and can be displayed on the in-built web browser. **Client-based** applications use the *WebView* class (Ogden, 2012) to embed web contents in the application. As smartphones are sold with a set of pre-installed root certificates, it is easier to trick users to accept spoofed SSL certificates on client-based applications.

As an example, consider the following scenario: To intercept traffic to and from Alice's smartphone, Eve decides to carry out an MiTM attack. With the help of social engineering, Eve is able to find out Alice's favourite games and tricks her to install a free application. However, to use this application, Alice is required to install a (rogue) SSL certificate generated by Eve; once the certificate is on Alice's smartphone, Eve can capture all the traffic from by the device – as shown in Figure 1.

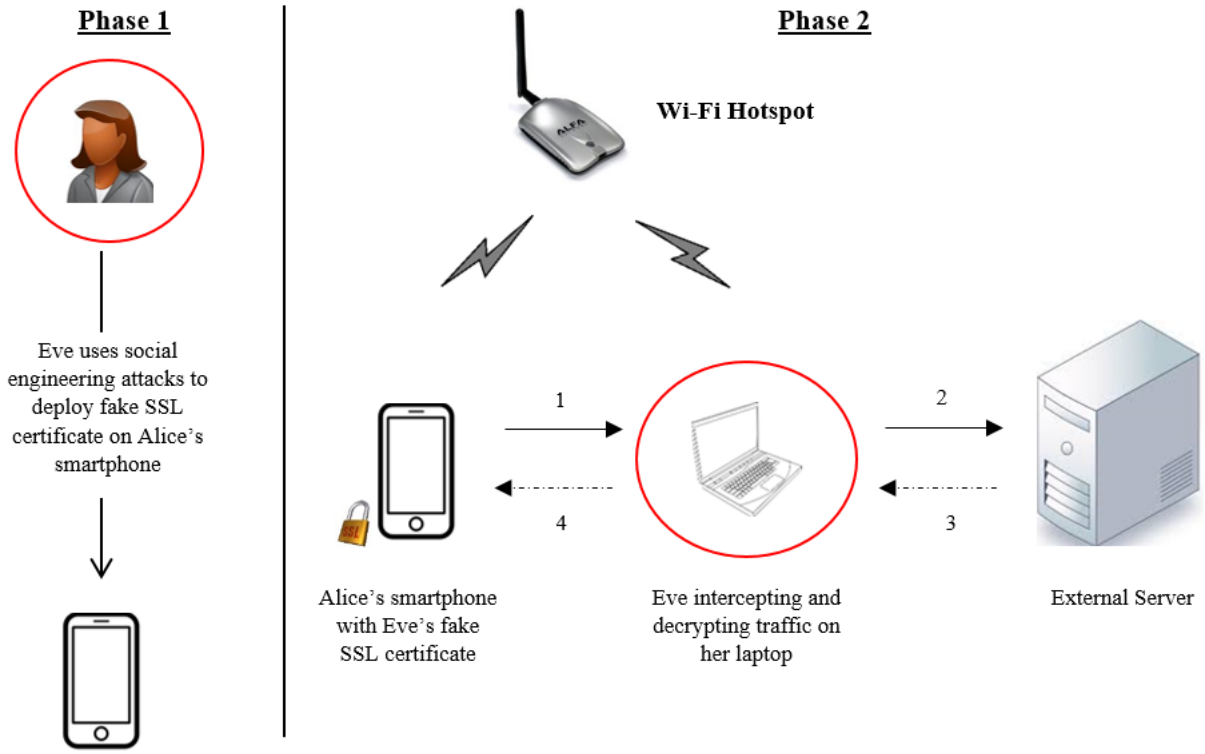

Figure 1: MiTM attack by SSL Hijacking

The fake SSL certificate allows Eve to intercept traffic generated by both web- and client-based applications. If an application on Alice's smartphone needs to connect to an external server over the web, a DNS query will be performed to establish the connection. From this point onwards, Eve's computer will act as the (MiTM) proxy and capture packets and relay them to and from Alice's smartphone and external servers.

SSL Stripping

The concept of SSL Stripping was first introduced in 2009 (Marlinspike, 2009). The main idea behind this attack is to give the victims a false sense of security by believing that they are communicating over an HTTPS connection, when in fact the MiTM attack will relay traffic over an HTTP connection.

It has been observed by Marlinspike in (Marlinspike, 2009) that smartphone users rarely type the URL in its entirety, instead only inputting the domain name part of a website. Normally, the application then proceeds by redirecting the traffic over an HTTPS connection. SSL Strip intercepts the HTTPS redirect and maps the links to its HTTP equivalent. While the attacker will still communicate to the server over an HTTPS connection, the client on the other hand will receive traffic over an HTTP connection and will be unaware of it, as depicted in Figure 2.

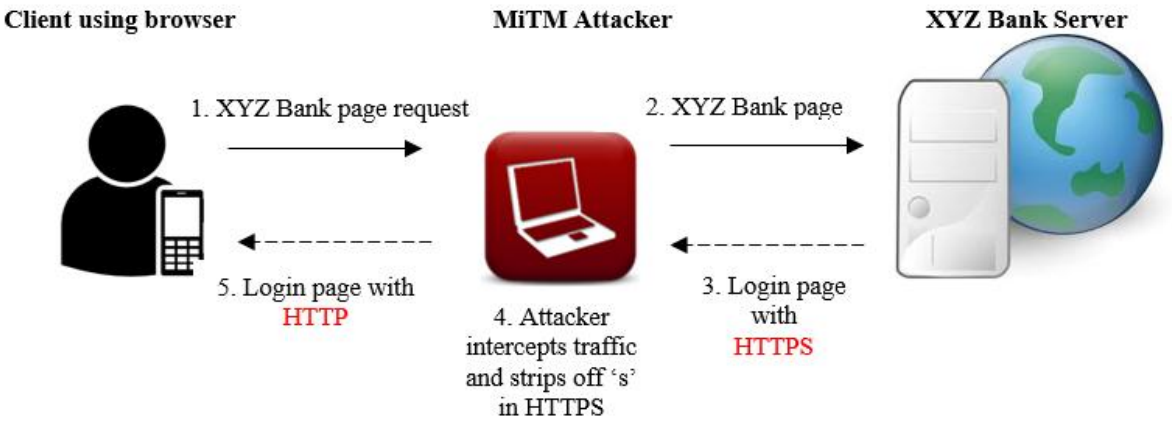

Figure 2: MiTM attack by SSL Stripping

DNS Spoofing
This type of MiTM attack targets the DNS protocol. DNS translates logical web addresses into their corresponding IP addresses. Every DNS query that is sent over the network contains a uniquely generated identification (ID) number, whose purpose is to identify queries and responses. To carry out DNS Spoofing, the attacker can intercept a DNS query sent out by the client and extract the unique ID to create a fake DNS response which will be accepted by the client. Currently, such attacks cannot be easily detected on the smartphone, hence, making the attacker's job easier to re-route traffic to a malicious server.

In the remainder of the paper, we provide further discussion on SSL and DNS and explain how to mitigate the attacks mentioned here by applying SSL Pinning and/or DNSSec.

RELATED WORK

The authors of (Fahl, et al., 2012) studied incorrect implementations of SSL in Android applications. Their proposed tool revealed that 8% of the 13,500 applications considered included SSL code that rendered the applications prone to MiTM attacks. For those applications where the MiTM attacks were successful, Fahl et al. collected credentials for applications belonging to categories including social networking, finance and business. Additionally, the authors were able to tamper with the virus signature database of an antivirus application in order to inject fake signatures or even completely disable the application. In (Fahl, Harbach, Perl, Koetter, & Smith, 2013), the authors conducted a survey, involving application developers, to determine the likely causes of incorrect SSL implementation on the smartphone platform. Based on the findings, a new, simplified scheme for handling SSL was proposed.

The work of (Tendulkar & Enck, 2014) and of (Clark & van Oorschot, 2013) investigated the challenges faced by application developers when implementing SSL and performing SSL verification in smartphone applications. Both papers highlighted the usage of SSL Pinning as a means to mitigate MiTM attacks. However, the authors also pointed out that due to lack of documentation, application developers often misuse SSL P inning and as a result, the applications are rendered more vulnerable to MiTM attacks.

We turn next to discussions of SSL Pinning and DNSSec.

SSL PINNING

On the desktop platform, web browsers are the default option for accessing contents over the Internet. However, in smartphone applications, there exists a limited number of web browsers, which in turn makes it easier to embed all the necessary controls to validate SSL certificates properly. On the other hand, the smartphone platform is application-centric and applications can be written by anyone who owns a developer's account. Whilst each platform has strict application development guidelines, developers do not always abide by the rules and the application source code is, in general, not verified for correct SSL implementation before being uploaded to the marketplace. Similarly, there are several other factors, as described in (Fahl, Harbach, Perl, Koetter, & Smith, 2013), that can lead to improper use of SSL while building an application.

However, most of these problems can be eliminated by the use of **SSL Pinning** which ensures that the application checks the server's certificate against a known copy bundled in the application before it is deployed on the market. An application developer knows in advance which servers the application is required to connect to and, in turn, can specify in the source code the certificates that should be trusted.

Generally, as in web browsers, smartphone platforms are despatched with a set of trusted root CA certificates. For instance, Android 4.2 and iOS 6 each trust more than 100 root CA certificates. Instead of blindly trusting all the embedded root CA certificates, SSL Pinning allows the developer to create their own *Keystore* which includes their trusted certificates; when requesting acceptance of a certificate, the application will use the customised *KeyStore* to check for the server certificate and hostname before proceeding with transfer of information.

While it is an attractive and simple way of mitigating MiTM attacks, SSL Pinning is not completely immune to vulnerabilities. As demonstrated in (Andzakovic, 2014), SSL Pinning can be disabled by reverse engineering the application and forcing it to accept spoofed SSL certificates. Additionally, the use of third-party advertising libraries can impact on the effectiveness of SSL Pinning. Application developers are required to consent to the use of the certificates provided by the advertising companies or they might not earn any revenue from advertisements. Since there is no control on the trusted root CA certificates or intermediate certificates used for advertising purposes, the application can be easily enticed into accepting spoofed certificates sent by attackers.

SSL Pinning relies heavily on its correct implementation by the developers, and so is prone to human error. Bugs may remain hidden in the source code or temporary testing code left behind after the application is deployed.

In the remainder of the paper, we focus on the DNS and investigate the use of DNSSec, which requires minimal developer involvement, as a means to thwart MiTM attacks. We believe that correct implementation of DNSSec would provide a greater level of protection against MiTM attacks than would SSL Pinning.

DNS & DNSSEC

Overview of DNS

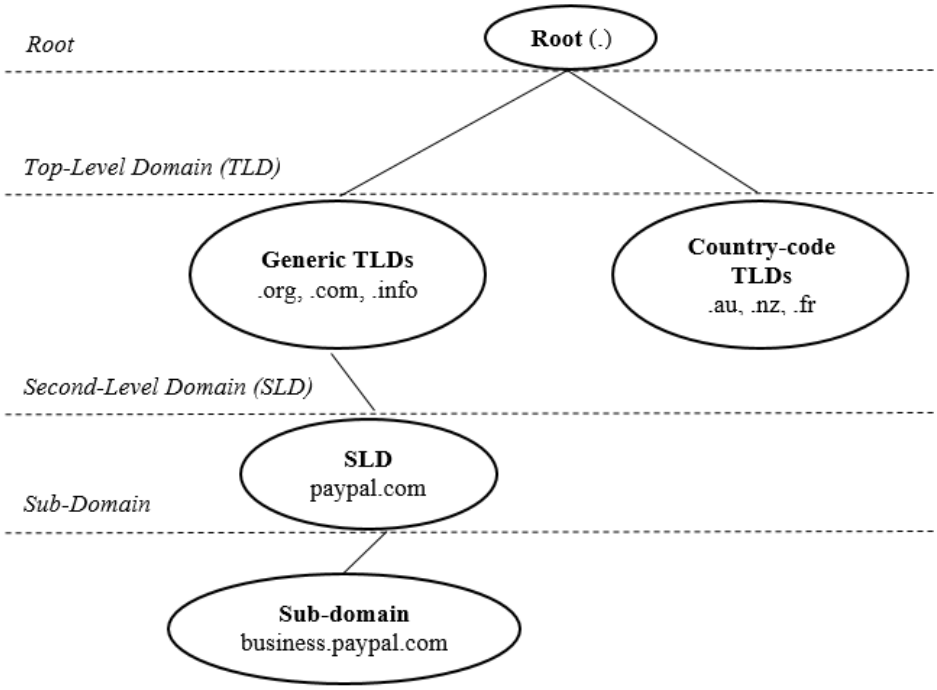

Figure 3: DNS Domain Structure

The Domain Name System (DNS) is a query mechanism which links logical names (e.g. www.google.com) to IP addresses (e.g. 74.125.131.105). It was initially proposed in the 1980s and the details of its implementation are described in the standard RFC 1035 (Mockapetris, 1987). The DNS process can be divided into 4 steps: (i) Delegation, (ii) Zone File Management, (iii) Zone File Propagation and (iv) Resolving.

DNS **Delegation** follows an inverted tree structure, as depicted in Figure 3, and is used for name resolution. The structure starts with the root and is represented by a dot. Clients resolve names by following a chain of authoritative servers (also referred to as name servers), starting from the Root, followed by the Top-Level Domain (TLD) name servers, Second-Level Domain (SLD) name servers until the queried logical name is found. For example, *business.paypal.com* is a sub-domain of the SLD *paypal.com*, which in turn is a sub-domain of the generic TLD *com*, located under the Root domain.

A zone may contain information about a domain and its sub-domains. Each piece of information about the domain, such as *Owner Name* (www.paypal.com), *Class IN* (Internet), *Type A* (IPv4 address) are contained in a Resource Record (RR), which is recorded as a single line in a zone file. We refer the reader to (Linux, 2003) for more information on zone files. The set of all RRs associated with the same owner name, class and type is called the Resource Records Set (RRSet). The primary purpose of the zone file is to link a domain to an IP address. **Zone File Management** is performed by DNS operators who are responsible for looking after particular zones. The RR includes other information, including Time To Live (TTL), which indicates the length of time it can be stored in cache and used for dissemination. This process is known as the **Zone File Propagation**. The entire process of answering a DNS query is known as **Resolving**.

Since no authentication and integrity checks are done during resolving, it provides a very convenient opportunity for attackers to divert traffic via their MiTM proxies as DNS servers are unable to verify the correctness of incoming DNS data. This is the main problem with the DNS set-up, for which DNSSec was proposed as a solution. In the next section, we explore the application of DNSSec to protect DNS servers and prevent MiTM attacks.

Overview of DNSSec

To compensate for the lack of security in DNS, DNSSec - was implemented and standardised by the Internet Engineering Task Force in 1997 (Kaufman, 1997). DNSSec can be considered to be a set of security extensions added to the existing DNS protocol and it relies on the 'chain of trust'. In fact, the extensions only affect the **Zone File** (second and third step) in the DNS process while the first (**Delegation**) and last (**Resolving**) steps remain unchanged. We provide a comparison of DNS and DNSSec in Table 1.

Table 1: Comparison of DNS and DNSSec

DNS	DNSSec
1. Delegation	1. Delegation
2. Zone File Management	2. Zone File Management
3. Zone File Propagation	*3. Zone File Signing (RRSIG and DNSKEY)*
4. Resolving	4. Zone File Propagation
	5. Resolving
	6. Verifying

DNSSec applies the concept of Public Key Cryptography to authenticate the origin of data and data integrity, and to verify the 'non-existence' of a domain (also referred to as authenticated denial of existence). For each domain, a key pair, consisting of a public key and private key, is generated and provided by the zone administrator. The private key is used to sign each RRSet in a zone file. As a result, a digital signature is computed and is referred to as the *Resource Record Signature* (RRSIG). The public key, also known as DNSKEY, and its corresponding digital signature (RRSIG DNSKEY) are also added to the zone file. Finally, the *Next Secure* (NSEC) record together with its signature (RRSIG NSEC) are included to detect the 'non-existence' of domain names. The above process is known as the *Zone File Signing* and is listed as Step 3 in Table 1.

To verify the authenticity of the DNS data (Step 6 from Table 1), the DNSKEY of the sub-domain is verified against a copy stored at the SLD. In turn, a copy of the SLD's DNSKEY is verified at the TLD and the TLD's DNSKEY is verified by the Root zone. This 'chain of trust' provides assurance that the signatures and keys are correct and the DNS data is authentic.

In the event of an MiTM attack, the client application will be unable to verify the DNS data from the MiTM proxy and the traffic will be interrupted because the smartphone client will not accept traffic that has been rerouted from the MiTM proxy which is controlled by the attacker.

Table 2 provides a list of attacks which can be thwarted using DNSSec.

Table 2: Mitigating attacks using SSL Pinning and DNSSec

Types of Attacks	SSL Pinning	DNSSec
SSL Hijacking	No	Yes
SSL Stripping	Sometimes	Yes
DNS Spoofing	No	Yes

DISCUSSION

Although, DNSSec has been standardised since 1997, there has been a slow adoption of the technology. According to Internet Corporation for Assigned Names and Numbers, there exist 627 TLDs and only 70% of them have been signed as of June 2014. We highlight some of the challenges and limitations of DNSSec.

Challenges

Currently, there exist no incentives for smartphone developers to use DNSSec as a security measure for their applications. This is partly due to the lack of attention the protocol has received since its introduction but also due to the cost since "bandwidth and storage requirements are increased by about a factor of 6 over DNS" (Fetzer & Jim, 2004). In addition, the authors of (Pfeifer, Fetzer, & Jim, 2005) show "how the most important advantages of the different proposals for enhanced DNS transaction security can be reached using existing infrastructures and technologies … it provides the advantages of asymmetric cryptography and reduces the configuration effort."

At present, the smartphone platform is dominated by four popular OS. Implementation of SSL Pinning varies depending on the host OS and it can be time-consuming to verify its correctness during the application vetting process; with DNSSec, application analysts can simply extract the server domain name and check whether it is DNSSec-enabled.

Furthermore, while smartphone users expect their applications to be secured, they have limited knowledge on how to mitigate SSL-based MiTM attacks. For instance, in some cases, attacks using SSL Stripping can be thwarted by simply typing the full HTTPS URL into a browser, instead of only the domain name.

Additionally, the amount of software that allows implementation of DNSSec on DNS servers is limited. The most popular software, BIND, provides an extensive range of functionalities for upgrading to DNSSec; however, many application developers do not have this knowledge. Another challenge with DNSSec is the use of algorithms for signing keys. Since the underlying concept of the protocol is based on 'chain of trust', all the zones on the bottom layers all the way up to the Root zone have to use the same algorithm. Any discrepancy along the way could invalidate the verification of keys as the digital signatures will not match.

Recommendations

To encourage smartphone developers to adopt DNSSec during the application development process, we believe smartphone OS owners should request a form of certification from the developers before their applications can be uploaded on the application market. This certification will provide a guarantee that the developer has correctly implemented SSL Pinning and/or DNSSec and that security checks have been implemented correctly.

There is also a need to continuously educate smartphone users on how to take cautionary measures to mitigate MiTM attacks. Given that antivirus applications offer no protection against these types of attacks, users should be made aware of the implications of accepting SSL certificates during application installation and of connecting to free public WiFi networks. When using the browser on their smartphones, users are advised to consistently type an HTTPS URL in full to reduce the chances for an attacker to successfully apply SSL Stripping. Although

URLs can be fairly lengthy, smartphones allow users to add keywords to the in-built dictionary; thus simplifying future HTTPS URL entries.

CONCLUSION

The exponential surge of smartphone data traffic is providing attackers with an avenue by which to deploy MiTM attacks easily. In this paper, we described three SSL- and DNS-based MiTM attacks and explained how they can be mitigated using SSL Pinning and DNSSec. Current observations highlight a lack of user awareness about MiTM attacks on smartphones and so we outlined the challenges and limitations as to why smartphone users, application developers and application market hosts fail to prevent such attacks. Finally, we presented some recommendations which can encourage the adoption of techniques such as SSL Pinning and DNSSec to thwart MiTM attacks.

REFERENCES

Andzakovic, D. (2014, May). *Bypassing SSL Pinning on Android via Reverse Engineering* . Retrieved from Security-assessment.com: http://packetstorm.interhost.co.il/papers/general/android-sslpinning.pdf

Clark, J., & van Oorschot, P. C. (2013). SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. *Proceedings of 2013 IEEE Symposium on Security and Privacy*, (pp. 1-15). California.

Codenomicon. (2014, April). *The Heartbleed Bug*. Retrieved from HeartBleed Bug: http://heartbleed.com/

Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., & Smith, M. (2012). Why Eve and Mallory love Android: an analysis of Android SSL (in)security. *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 2012)*, (pp. 50-61). North Carolina.

Fahl, S., Harbach, M., Perl, H., Koetter, M., & Smith, M. (2013). Rethinking SSL Development in an Appified World. *Proceedings of the 20th ACM Conference on Computer and Communications Security (ACM CCS 2013)*, (pp. 1-12). Berlin.

Fetzer, C., & Jim, T. (2004). *Incentives and Disincentives for DNSSEC Deployment.*

IDC. (2014, January). *Worldwide Smartphone Shipments Top One Billion Units for the First Time, According to IDC*. Retrieved from IDC: http://www.idc.com/getdoc.jsp?containerId=prUS24645514

Kaufman, C. (1997, January). *Domain Name System Security Extensions*. Retrieved from http://tools.ietf.org/html/rfc2065

Linux. (2003). *Zone Files*. Retrieved from Linux Reference Guide: BIND: http://tiny.cc/punshx

Marlinspike, M. (2009). *SSL Strip*. Retrieved from http://www.thoughtcrime.org/software/sslstrip/

Mockapetris, P. (1987, November). *Domain Names - Implementation and Specification*. Retrieved from http://www.ietf.org/rfc/rfc1035.txt

Ogden, M. (2012, September). *Building WebView Applications*. Retrieved from http://maxogden.com/building-webview-applications.html

Pfeifer, G., Fetzer, C., & Jim, T. (2005). Using SSL For Secure Domain Name System (DNS) Transactions. *Transactions of the Technical University of Ostrava, No.2*, pp.101-106.

Richter, F. (2013, November). *Global Smartphone Traffic to Increase 10X by 2019* . Retrieved from Statista: http://www.statista.com/chart/1617/global-smartphone-traffic/

Schwartz, M. J. (2014, February). *Apple SSL Vulnerability: 6 Facts*. Retrieved from Dark Reading: http://www.darkreading.com/vulnerabilities-and-threats/apple-ssl-vulnerability-6-facts/d/d-id/1113992

Tendulkar, V., & Enck, W. (2014). An Application Package Configuration Approach to Mitigating Android SSL Vulnerabilities. *Proceedings of the 2014 Mobile Security Technologies (MoST 2014)*, (pp. 1-10). California.