Early Detection of forged/misissued certificates and some loopholes in our existing Revocation schemes

A Survey Report

Abhishek Kejriwal *Rahul Sinha*

Abstract

One of the main issues in the current PKI is the absence of any effective mechanisms for early detection of forged/misissued or fraudulent certificates and thereby revoking them. In our report we will discuss about the current PKI infrastructure, the existing mechanisms for such early detection and their ineffectiveness. We then discuss how our existing revocation schemes also add to the cause of SSL end users being compromised by a revoked certificate. Finally we discuss about the Certificate Transparency framework currently being developed by Google, which is an open framework for monitoring and auditing digital certificates in the current PKI model and thereby aiming to address to the problem of early detection of malicious certificates.

1. Introduction

Back in the 1970's, it was only the Symmetric Key Infrastructure that was known to the people for having a secure and confidential communication over the network. The symmetric key technique involved the use of a ***private key*** which served as an important feature for maintaining the integrity and authenticity of the communication. This methodology proved inefficient in cases where the two parties who wanted to have a secure communication were unknown to each other, and this slowly and gradually proved to be a big problem. This led to the establishment of a new architecture consisting of a central server known as the Key Distribution Center (KDC). Each entity on the network shared a secret key with this Key Distribution Center. The number of secret keys in the KDC was equal to the number of entities in the network. The KDC served as a single point of contact for two unknown entities who wanted to have a secure communication. But, soon it was realized that using the concept of central server for introducing two unknown entities was not an efficient solution because of the following two reasons:

1. The central server i.e. the KDC would always have to be kept online, and
2. A single point of failure in the KDC would result in an insecure and lack of confidential communication between the unknown entities

It was then when two renowned researchers, Diffie and Hellman came up with a new concept of Asymmetric Key Infrastructure where a key pair was generated between the two communicating entities instead of a single secret key as in symmetric key infrastructure. The key pair consisted of a public key - which was published by the communicating parties in the network and individual private key for both the entities. This eventually lead to the emergence of ***Public Key Infrastructure***.

2. Public Key Infrastructure (PKI)

In the cryptographic world, the process of associating a public key with an entity by means of a ***Certificate Authority(CA)*** is described as a Public Key Infrastructure. In general a PKI involves the active participation of hardware, software, services and policies to enact together to provide authentic and confidential communication to a client and a server over the network by making appropriate and effective use of digital signatures and the services provided by a CA. The PKI provides important documents in the form of digital certificates which are necessary for a client and a sever who want to establish a secure ***SSL/TLS*** connection. Some of the key components of a PKI are:

1. Certificate Authority.
2. Registration Authority.
3. Digital Certificates.

4. Domain owners
5. End users.

In this report we will focus on the issues with the current PKI related to early detection of malicious certificates, existing solutions and their ineffectiveness, problems introduced by the current revocation schemes and how Certificate Transparency framework aims to address them.
We now define some key terms and their definitions in context of PKI.

2.1 Certificate Authority

Certificate Authorities are entities in the PKI who are responsible for issuing X.509 certificates to domain owners and they are also responsible for certificate revocation. The Root CA is a CA whose trust is assumed. The certificates of the intermediate CAs are issued and signed by the Root CA.

2.2 Certificate Revocation

Certificates issued by a CA have specific expiration dates after which the use of the certificate is not desired. A new certificate is required to be purchased by the domain owner again after expiration. A certificate may be revoked even before its actual expiry. The major reason behind the certificates being revoked is the loss or misuse of the private key of the certificate issuing entity or if the certificate issued by the CA is compromised or mistakenly signed i.e. issue of a fraudulent certificate. DigiNotar was declared bankrupt on September 2011 for being involved in issuing fraudulent certificates and its certificate was revoked causing many websites to fail. Two main mechanisms for certificate revocation are CRLS and OCSP, which we will discuss in the later half of the report.

2.3 PKI Risks

Even though PKI is the most successful and vastly followed practice to ensure a secure and authentic communication it bears some concerned risks. The ***protection of private key*** is not assured because if the private key of the end user is compromised then it can be used by the adversary to sign the messages sent by the end user. If the private key of the CA is compromised then it may result in the issuance of fraudulent certificates. The ***trust factor*** which is an important characteristic of the PKI can also be questioned if any of the private key is compromised. If the adversary imposters the end user to sign the messages then the characteristic of non repudiation is questioned.

3. Secure Socket Layer(SSL)

The SSL or the Secure Socket Layer is that protocol which serves as the bottom layer for the Secure Hypertext Transfer Protocol(HTTPS). The SSL is basically a communication protocol which provides secure and authentic communication to the entities communicating over the internet. The SSL derives its security features from the various services offered by the Public Key Infrastructure (PKI) embodied in the form of Certificate Authority(CA) and Digital Certificates. There are various steps that SSL carries out to provide a secure and confidential exchange of information and data, but in our report we will only talk about the steps involved in the handshake process and the security and risk factors associated with it. The handshake process is the beginning of the secure connection between the server and the client. The most important messages exchanged to complete the handshake process includes - ClientHelloMessage, ServerHelloMessage, KeyExchangeMessage and ChangeCipherSpecMessage.The client first sends the ClientHelloMessage enclosing the various ciphers available to the client along with a random number generated by the client. The server then sends a ServerHelloMessage to the client stating all the cipher suites supported by the server along with a server generated random number. The server also sends a digital certificate consisting of the servers public key signed by a Certificate Authority. Now the client sends a ClientKeyExchangeMessage which consists of the sessions key that would be used by the server and the client for secure communication. Finally the ChangeCipherSpecMessage is exchanged between the server and

the client stating that the particular session would be encrypted using the derived sessions key. Now since we know that SSL acts as a key protocol to identify the identity of the server sending the digital certificates, the SSL certificates are often signed by intermediated authorities rather than the trusted root CA. The client on receiving the certificate should look for entire chain of the authorities signing that particular certificate and the sign should end with root CA at the top.

4. SSL Man in the Middle Attack(MITM)

Although, SSL protocol was devised to provide for a secure connection to enable confidential communication between the entities, it is highly vulnerable to Man in the Middle attacks. The MITM attack takes place if the adversary is successful in convincing the client that the entity to whom the client is talking to is a legitimate identity. The adversary places himself in middle of the client(browser) and the server(web server). Whenever the client wants to talk to the server the client opens an active SSL connection. The adversary if successful in intercepting the connection responds to the request with a message along with a certificate which signs the adversaries public key. The client on accepting the certificate is now compromised. The adversary simultaneously opens a connection with the server and acts as a client with respect to the server. The client and server assumes that they have only one active SSL connection but in-fact the presence of adversary leads to two active connections thereby allowing the adversary to monitor the communication between the two entities. This may result in severe consequences to the client in the way that the adversary can intercept the clients password and modify the code. There are several variations in which the adversary can carry out these MITM attacks. The attacker can set up a malicious router which may appear to be legitimate and then he may force the user to connect to that router. For example, the adversary can set up a Wifi hotspot by a common name such as a one in a cafe and can coerce the users to connect to that hotspot. As soon as the users connect to that hotspot the attacker can quietly and silently eavesdrop on the sessions between the two entities.

5. Current mechanisms for protection against mis issued/fraudulent certificates, from the end users perspective.

After having understood the characteristics and working of the Secure Socket Layer and the security breaches that SSL can be exposed to, its essential to protect the end users from being compromised by a malicious certificate received at the end of SSL handshake. Here are a few existing mitigation techniques and their trade-offs.

5.1 HTTP Strict Transport Security(HSTS)

The HTTP strict Transport Security or HSTS was proposed by Hodges, Jackson and Barth in the early 2008. According to the HSTS, the servers made mandatory for the clients to communicate over HTTPS connections only. Moreover, the implementation was such that any certificate errors including TLS errors resulted in failing of the connection without giving the end users the leverage to ignore the insecure warning. It also recommended that the browsers should disable the loading of resources that were insecure on the HSTS enabled web pages.

5.2 Public Key Pinning(PKP)

In Public Key Pinning with HTTP extension, the server instructs the browser to accept only those certificates which have the servers public key pinned in the HTTP header and reject any certificate with unknown public keys. So whenever a SSL handshake is done, the browser compares the pinned public key of the server in the HTTP header, with the public key in the certificate which it received after the SSL handshake and rejects it if it differs.

5.3 Trusted Assertions for Certificate Key(TACK)

The mechanism of Public Key Pinning suffers from a shortcoming, where the server is limited to send the same certificate for the same web application in order for the end user to approve the authentication and legitimacy

when compared with the pinned value. TACK offers a comprehensive solution to this problem, by allowing the server to choose a key known as the TACK key which is used to sign the servers TLS keys. Now, whenever a client establishes a connection with the server, the server sends its TSK. Once the client receives the TSK for multiple times for the same host, that particular TSK is activated and the client activates that pin for a particular period of time. Every time the client connects to the same server for any service, it will require the connection to be signed with the same TACK key or else the connection would be rejected. In this way the client need not trust the certificate and the server's limit to deploy the same certificate for every new connection is also lifted.

5.4 Perspectives

The launch of the concept of Perspectives form Mozilla added a new direction in the research to prevent the users from getting trapped in the mitm attacks. Until now, the users have been relying on the so called trusted Certificate Authorities for a valid certificate from a server running a web application. Perspectives added a decentralized 3rd party called ***network notaries*** in the infrastructure. Any server in the network could be configured to become a network notary. The users would now have the freedom to trust any notary spread over the internet for certificate validation. The process works as follows:

a. *Once a notary is established the function of the notary is to constantly monitor the various websites and keep a record of all the certificates used by the websites till date.*
b. *The client on establishing a successful connection with the server, receives the server certificate.*
c. *The client then verifies the certificate for legitimacy in the cache. If there is any question regarding the trust factor of the certificate, the client contacts any of the trusted notary for their services.*
d. *The notary looks for any inconsistency in the certificate and replies to the client accordingly.*

In this way Perspectives make efficient use of the notaries to provide a decentralized service.

5.5 Convergence

Convergence is an extended version of Perspectives. Convergence is different from perspective in a manner because in Perspective the network notary responsible for validating the certificates can get hold of the end users IP Address and retrieve a full record of the users browsing history, but the anonymous nature of Convergence i.e. the ability to shield the IP Address of the end user from the notary and preventing misuse of any relevant information is the feature that makes Convergence different from Perspectives.

5.6 Direct Validation Certificate (DVCert)

Even though the concept of perspective sounds refreshing, it suffers from a major flaw. According to the model of the Perspectives, any server can be configured to become a network notary. Hence, the model allows any adversary to become a network notary. This gives advantage to the adversary to establish a false positive for the end user making him to validate a fraudulent certificate. Moreover, existing trusted notaries can be compromised by the adversary resulting in the same after effect. DVCert proposes a rather effective solution to the end user to prevent itself from falling prey to the mitm attack. The DVCert makes use of the already existing Password Authenticated Key Encryption(PAKE) protocol. According to the PAKE protocol the client and server establish a shared secret key based on the user password. These shared secret are taken into use by the DVCert to attest the certificate. On receiving the certificate for the first time from the server the client retrieves all the latest certificate and shared information. The web application maintains a Domain Certificate List(DCL) data structure to store all the certificates issued to the client. The DCL includes the web applications server signed certificates along with all other 3rd party issued certificates issued for a particular session. The browser now stores the DCL temporarily in its cache and every time the browser requests for a connection it uses the certificates in the storage to compare them from all the certificates maintained in the DCL in order to decide for trustworthiness. If the certificate do not match with the certificates listed in the DCL the connection is refused. A new DVCert transaction is exceuted on expiration of the DCL.

5.7 DNS-Based Authentication of named entities(DANE)

The DNS-Based Authentication of named entities relies significantly on the DNS Security Extensions(DNSSEC). The DNSSEC provides an important feature that a key associated with a domain name can only be signed by the parent domain and no 3rd party trusted authorities. For example, the keys for "abcd.com" can be signed only by the keys for "com" and only the DNS root has the permission to sign the key for "com". In this way DNSSEC ensures that client gets the certificate issued and signed by the domain rather than any trusted Certificate Authorities. DANE allows the user to use the DNSSEC infrastructure. The most important characteristic for DANE is that the domain who is responsible to answer the question regarding the query for domain name is itself responsible for signing and managing the keys for that particular sub-domain or domain i.e. the DANE binds the public key data to the domain name.

5.8 Certificate Authority Authorization(CAA)

In the present scenario the web applications get their certificates form various Certificate Authority that are present all over the internet. One major issue with this implementation is that it often results in mis-issued certificates. Certificate Authority Authorization has been presented as a solution to this issue of preventing the CA to misissue certificate. The mechanism is such that it allows the DNS domain server to make a list of all the CA's that have the authority to issue a certificate for that particular domain.To set the CA preference the DNS maintains CAA records. For example, if the DNS wants CA1 to issue a certificate to the domain it would set CA1 in its domain records. Now suppose some external user requests CA2 to issue a certificate for the same domain, CA2 would first read the CAA record to see if it is listed in the preferences of the domain. If CA2 is not listed in the CAA records, it would warn the user of its absence from the record and at the same time send a notification to the domain owner making it aware of the fact that a certificate was requested for its domain by an external user. It is then upon the domain owner to decide whether to add CA2 to the records or prohibit the issue of any such certificate. The protocol has a few disadvantages.

a. *All CAs may not wish to participate in following the CAA protocol and may go ahead with issuing a certificate to an unknown adversary.*

b. *The entire implementation of CAA depends on DNSSEC, which is still not deployed widely.*

From all the above mitigation techniques we can derive an important conclusion that all these various ways offer a solution from the end users perspective thereby preventing the end user from falling prey to fraudulent certificates. But data from recent security breaches reflects that incidents of issuing fraudulent and mis-issued certificates have seen a sharp rise. The infrastructure needs some imminent solutions for early detection of such forged/misissued or fraudulent certificates.

6. How our current revocation mechanisms add to the cause of SSL users falling prey to Revoked Certificates.

6.1 Certificate Revocation.

Certificate revocation may be defined as the invalidation of association between the public key and the attributes embodied in a digital certificate. Whenever a certificate is issued to a party by a CA or a delegated CA, the certificate has some attributes such as the issuer name, issued to, issue date, expiry date, the public key of the party to whom the certificate is issued. A certificate may be revoked before its actual expiry date because of several reasons such as,

1. *A CA has improperly or mistakenly issued a digital certificate.*
2. *A private key might have been compromised.*
3. *Most common reason is the user no longer being in possession of the private key in question.*

So whenever an SSL/TLS client receives a certificate during an SSL/TLS handshake, the client first checks if the certificate is expired. If the certificate is expired then the client does not accept the certificate. But if the certificate is not expired, the client has an additional check to do. The client needs to check if the non-expired certificate it received has been revoked by some certificate revocation mechanisms which will be discussed next. If the certificates status is revoked, the certificate should not be accepted by the client as that certificate can no longer be trusted.

6.2 Current problems with Certificate Revocation

The main problem with todays PKI and certificate revocation is that there is no perfect solution which addresses both the timeliness and performance requirements of all parties. Moreover, there is a lack of mechanisms for early detection of improperly or mistakenly issued certificate by a CA, and thereby revoking the certificate. We will describe some certificate revocation mechanisms which will give a better understanding of the issues highlighted above.

6.3 Certificate Revocation List(CRL)

CRLs are the most common approach to revoking certificates. A CRL (issued by a CA who has issued the certificate in question to the owner in the first place) contains the list of unexpired certificates that have been revoked since the last CRL was published and also contains the time in the future when the next CRL will be issued. CRLs are issued periodically, say daily, weekly, monthly. Some of the tradeoffs related to using CRL are as follows:

1. CRLs are generally very huge. It is a burden on the network CRLs being pushed down by the CA every now and then. This is the main reason CRLs are published on a periodic basis and not every time a client wants to verify the status of a certificate it received.
2. CRLs fail to provide real time revocation status of a certificate in question. This is the biggest problem and some revocation schemes(like OCSP) have been developed to address this problem which we will discuss later in this paper.

Let us take an example to understand the second problem associated with using CRL as pointed out above. Let X be a CA who has issued the digital certificate for Alice and X publishes its CRL every day at say 5 pm. Suppose that X has revoked Alices certificate on Monday 6pm (one hour after X published its last CRL). Now say, at 7pm on Monday, Alice wants to do a transaction with Bob for which Alice produces her digital certificate containing her public key to Bob. Bob checks if Alices digital certificate is expired or not. If not then Bob checks if her certificate is enumerated on the last issued CRL by X at 5pm on Monday. Since her certificate was revoked after the last publish time, Bob will not find any entry related to Alices certificate in the CRL and will assume that the certificate is trustworthy and hence accept the certificate. So we can see the problem here. During this time gap of 24 hours, from Monday 5pm to Tuesday 5pm, Alice can go around producing her revoked certificate to other parties who will go ahead and accept the certificate.

6.4 Online Certificate Status Protocol (OCSP)

OCSP is another Internet Protocol which is used to obtain the revocation status of a digital certificate and was designed to be used as a replacement for CRLs to overcome its shortcomings as described above. In this revocation scheme, an OCSP responder (a server typically maintained and run by a Certificate Issuer) is placed in the PKI model. Whenever a client receives a digital certificate, it sends out an OCSP status request with the certificate specifications to the OCSP responder and suspends the acceptance of the certificate till it gets a response from the OCSP responder. The OCSP responder sends an OCSP response back to the client saying the certificate being queried about is either good, revoked or unknown.

1. A good response indicates that the certificate being questioned about is good to be trusted and accepted and is not revoked at the time the OCSP responder queries its database about it.
2. A revoked response indicates that the certificate in question has either been revoked temporarily or permanently.

This response may also indicate that the CA has no record of ever having issued the certificate in question in the past. These kind of certificates are referred to as non-issued certificates.
3. An unknown response indicates that the OCSP responder does not have any information about the certificate being questioned about, may be because the request specifies an unrecognized issuer (CA) and not the one being serviced by the OCSP responder. This leaves the decision with the client whether it wants to try another source of Certificate Status information such as CRL (as a fallback option),etc.

6.4.1 Comparison of CRL with OCSP

***1**. As compared to CRLs, OCSP request and response messages are not a burden on the network (comparing it to the mere size of a huge CRL or even delta CRL).*
***2**. Unlike periodic CRLs, OCSP provides a way of getting a real time revocation status of the digital certificate in question.*
***3**. Unlike CRLs not much CPU power is required for parsing OCSP response by the client unlike in the case of CRLs, where parsing a CRL requires complex client side libraries.*

6.4.2 Comparison of CRL with OCSP

***1**. Availability of OCSP responder is one of the prime concerns in this revocation mechanism. A good certificate may not accepted by a client if the client is not able to get any response from the OCSP responder because of many reasons such as network failure, malfunctioning of the OCSP responder.*
***2**. OCSP responders are vulnerable to denial of service attacks, when overwhelmed with a deluge of OCSP request queries.*
***3**. It is vulnerable to replay attacks. Old good responses may be captured by an attacker and can be replayed even after the certificate is revoked but unexpired.*
***4**. Unlike CRLs where the CRLs are downloaded by the clients to their browser caches, in OCSP we need to take into account the time taken to get a response from the OCSP responder. So there is always a latency involved in this revocation scheme unlike CRL, where the CRLs are always available in local cache.*

6.5 Certificate Transparency(still in draft) as a solution to early detection of malicious certificates.

Certificate Transparency is an open framework for monitoring and auditing digital certificates in the current PKI model through a system of Certificate logs, monitors and auditors. Google has been working on this scheme currently. It will be wise to point out that CT is not a replacement for the current revocation mechanisms such as CRLs, OCSP, but it is aimed at providing a way for early detection of mistakenly/fraudulently issued certificates and then revoking them using the current revocation schemes and thereby also stopping MITM attacks caused by these forged certificates. So, CT logs themselves do not prevent certificates from being mistakenly or fraudulently issued by a CA, but they give the interested parties such as domain owners the privilege to early detect any such cases so that they can go ahead and revoke the certificate as soon as possible.

6.5.1 Goals of Certificate Transparency

Certificate transparency aims at making the issuance and existence of digital certificates open to audit/scrutiny by domain owners, CAs and end users. The goals are to:

***1**. Make it impossible or very difficult for a CA to issue a certificate for a domain without this certificate getting noticed by the domain owner.*
***2**. Provide an open framework which lets domain owners or CAs to determine whether certificates have been mistakenly or maliciously issued.*
***3**. Protect end users. If the first two points are met, the SSL/TLS end users are by default in a safe place.*

Certificate Transparency consists of 3 main parts:

I. Certificate Logs

Certificate logs are generally log servers that contain cryptographically assured, publicly audit-able, append only records of certificates that have ever been issued till date. Whenever a CA issues a certificate to a domain say www.example.com, the CA submits the certificate it has issued to the log server. Number of logs should not be very less and it should not be huge too (say less than 1000 worldwide) and each of these log severs could be independently operated by a CA, an ISP or any interested parties (like Google). Whenever a CA submits a certificate it has issued to a log server, the server responds back with a Signed Certificate Time-stamp (SCT) which is a proof that the CA has requested the log server to append a new certificate and that the log server has acknowledged it by saying Yes, I got the certificate. And I promise that the certificate will appear in my log within a certain time period. This time period is called the maximum merge delay (MMD), the time taken for a certificate to get reflected on the log.

II. Monitors

Monitors are basically servers which are maintained by CAs or domain owner themselves, and they continuously contact the log servers and check if any unexpected certificates for a domain appear on the log servers and also check for certificates which have unusual certificate extensions and permissions like that of a CA. Monitors will be run by domain owners such as Amazon, Google, or a bank, or any Government Organization. There will also be CAs or some third party service providers who will provide monitoring services for those customers who do not wish to set up their own monitors.

III. Auditors

Auditors may be an integrated component of the end users web browser and they perform two functions. They verify that log servers are behaving properly and they are consistent cryptographically. Whenever an end user receives a certificate after an SSL/TLS handshake, apart from the normal path validation process, now the auditor also checks if the certificate it received has been registered in a log or not. If the certificate has not been registered in any log, that certificate might have reasons for not being trusted.

Currently there are three supported methods for delivering an SCT with a certificate.

X.509v3 Extension

An issuing CA submits the pre-certificate to the log, and the log returns an SCT to the CA. Once it gets the SCT the CA then attaches the SCT to the pre-certificate it created before as an X.509v3 extension. It finally signs the certificate and delivers it to the server operator (the buyer/domain owner). No server modifications are required by the server operators if this method is used.

TLS Extension

In this method the CA certificate issuance process remains unchanged. The CA issues the certificate to the domain owner and the domain owner submits the issued certificate to the log server. The log server returns an SCT to the domain owner and the domain owner uses a TLS extension of type signed certificate time-stamp to deliver the SCT to an SSL/TLS client during SSL/TLS handshake.

OCSP stapling

In this method, the CA issues the certificate simultaneously to both the server operator and the log server. The server operator then sends an OCSP query to the CA and the CA responds back with the SCT. This method does not delay the issuance of the certificate as the CA can get the SCT from the log server asynchronously. The server operator once it has the SCT, can include the SCT in an OCSP extension and send it to the SSL/TLS client during the TLS handshake..

6.5.2 Working of Certificate Transparency

A CA creates a pre certificate which it needs to issue to a buyer and submits it to the log server prior to

everything. The log server responds back with an SCT (Signed certificate timestamp) for that certificate to the CA. The CA then includes this SCT in the certificate it is supposed to issue to the buyer (say in an X509v3 extension field). We can assume this buyer to be a domain owner such as www.example.com. Now whenever a SSL/TLS client performs an SSL handshake with www.exmaple.com, it receives the certificate having the SCT (provided by the log server in the first place). The existence of this SCT proves that the certificate has been logged in some log server. The auditor on the end user then queries the log server to check if the SCT has indeed been signed by that log server and also checks if the certificate has been legitimately added to this log server. If the results are unexpected, the client can decide to not trust the certificate it received and hence terminate the SSL connection to www.example.com. When Certificate Transparency gets deployed on a large scale, it is desirable not to trust a certificate which has no SCT, meaning it has not been logged in any log server.

6.5.3 Log Proofs

Certificate Transparency uses Merkle hash tree to facilitate public auditing of certificate and logs. A Merkle hash tree is a simple binary tree containing hashed leaves and nodes. Leaves are the hash of the individual certificates themselves and the nodes are the hashes of paired child leaves or paired child nodes. The root hash is known as the Merkle tree hash and this is advertised by the log server. The log server signs the Merkle tree hash with its private key and it is known as the signed tree head (STH). The log server appends all the new certificates periodically to the log. It does so by creating a separate Merkle tree hash with all the new certificates and then combining this new Merkle tree hash with the old Merkle tree hash to form a new Merkle tree hash. It is then signed to create a new signed tree head (STH). In this way the Merkle tree grows in size with new certificates being appended to the log.

6.5.4 Consistency Proof and Audit Proofs

Because of the unique structure of Merkle hash tree, it is able to provide two very important information known as the consistency proofs and the audit proofs that too very efficiently and quickly.

1. *Consistency Proof proves that the consistency of the log is being maintained even while appending new certificates to it. They are used by the monitors(domain owners) in general.*
2. *Audit Proof proves that a particular certificate is indeed present in a log. They are used by the auditors(on the SSL client browser) in general.*

6.5.5 Benefits of Certificate Transparency

The benefits are mainly aimed at certificate authorities, domain owners, and server operators, but they also affect individual users.

1. *Log servers provide a fantastic framework for monitoring and auditing any certificates issued till date.*
2. *If a CA goes rogue, it is very easy for domain owners to detect any illegitimate certificate issued for their domain using the monitor services provided by the CA or their own monitors.*
3. *Consistency Proofs and Audit Proofs provided by logs help determine if a log is behaving consistently or not, and whether a Certificate received by an end user is logged in one and hence legitimate.*
4. *With the use of SCTs it becomes very much easy for an end user browser to detect if a certificate is a legitimate one or not.*
5. *With all these in place, revocation can be done much early as the detection process is speeded up.*

7. Conclusion

So, in the current PKI infrastructure, we have a problem with early detection of forged/malicious certificates. The existing solutions discussed in this report are not fully effective for such early detection and thereby revoking them. Also, our revocation mechanisms are somewhat inefficient and in certain cases even after a certificate is revoked, users fall prey to forged/malicious certificate because of its absence in a CRL because of its periodic

nature. Also in the OCSP scheme, an end user may end up trusting a certificate as a result of a replay attack done by an adversary who had captured an old OCSP *Good* response and replayed it even after the certificate in question is revoked. So, keeping all these things in mind, we can say that the Certificate Transparency framework is an excellent idea not only for SSL end users but also domain owners and CAs for early detection of any malicious certificates. If we know early, we can initiate the revocation mechanism early even though we have pointed out that they are too inefficient is some ways. We conclude by saying, Detect early, revoke early.

References

[1]. Analyzing Forged SSL Certificates in the Wild, by Lin-Shung Huang, Alex Ricey, Erling Ellingseny, Collin Jackson

[2]. Ten Risks of PKI: What Youre not Being Told about Public Key Infrastructure, By Carl Ellison and Bruce Schneier.

[3]. Can We Eliminate Certificate Revocation Lists? By Ronald L.Rivest

[4]. A response to Can We Eliminate Certificate Revocation Lists?, by PatrickMcDaniel and Aviel Rubin

[5]. X.509 Forensics: Detecting and Localising the SSL/TLS MenintheMiddle by Ralph Holz, Thomas Riedmaier, NilsKammenhuber, Georg Carle

[6]. Efficient Certificate Revocation List Organization and Distribution, by Jason J. Haas, YihChun Hu, and Kenneth P.Laberteaux

[7]. A Distributed Online Certificate Status Protocol with a Single Public Key, by Satoshi Koga and Kouichi Sakurai

[8]. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements, by JeremyClark and Paul C. van Oorschot

[9]. Analysis of the HTTPS Certificate Ecosystem, by ZakirDurumeric, James Kasten, Michael Bailey, J. Alex Halderman

[10]. The Secure Sockets Layer (SSL) Protocol Version 3.0, Request for Comments: 6101, IETF

[11]. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Request forComments: 5280, IETF

[12]. Internet X.509 Public Key Infrastructure Online Certificate Status Protocol - OCSP, Request for Comments: 5912, IETF

[13]. Certificate Transparency, Request for Comments: 6962, IETF (In draft)

[14]. "Digital Certificate Revocation" by Sally Vandeven, Walter Goulet

[15]. "Trust No One Else: Detecting MITM Attacks Against SSL/TLSWithout Third-Parties" by Italo Dacosta, Mustaque Ahamad, and Patrick Traynor

[16]. "Upgrading HTTPS in mid-air: An empirical study of strict transport security and key pinning" by Michael Kranch, Joseph Bonneau

[17]. "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing" by Dan Wendlandt, David G. Andersen, Adrian Perrig

[18]. "PKI Basics - A Technical Perspective" by Shashi Kiran, Patricia Lareau, Steve Lloyd

[19]. "Reducing the X.509 Attack Surface with DNSSECs DANE" by Eric Osterweil, Burt Kaliski, Matt Larson, Danny McPherson

[20]. "https://tools.ietf.org/html/rfc6698" - DNS based Authentication of Named Entities

[21]. "https://tools.ietf.org/html/rfc6844" - DNS Certification Authority Authorization (CAA) Resource Record

[22]. http://perspectives-project.org/

[23]. http://tack.io/

[24]. http://www.certificate-transparency.org

[25]. www.google.com [26]. www.wikipedia.org [27]. www.youtube.com [28]. www.stackoverflow.com