SECURE E-COMMERCE

Research Work: Secure e-Commerce?
PKI and SSL/TLS certificates vulnerabilities

José Munera López - 100039406
Víctor Pimentel Rodríguez - 100055446
December 23th 2009

Index

SECURE E-COMMERCE? PKI AND SSL/ TLS CERTIFICATES VULNERABILITIES

José Munera López - 100039406
Víctor Pimentel Rodríguez - 100055446

Introduction

During the last few years we have witnessed a great and constant increase in the e-commerce users, volume of sales and supporters. One of the key points that made possible this incredible expansion of e-commerce was that users trust in the systems they use; it makes sense to think that few people would have make money transactions through a non reliable channel.

The security required by users, sellers and banks (to mention some entities involved) has been provided by a set of standards and protocols designed to make electronic communications safer and more reliable. These elements include SSL/TLS protocols and the PKI design, which are going to be the main subjects of our analysis.

This document will try to perform a brief analysis of some of the most notable vulnerabilities of the Public Key Infrastructure (PKI) design and the SSL/TLS protocols used when stabilising a secure connection, pointing out the importance of a responsible use of the eCommerce as the “last line of defence” against fraud or other electronic crimes.

Background

This section will try to introduce the reader to some of the knowledge necessary to understand this work.

Public Key Infrastructure

A Public Key Infrastructure (PKI) is a set of elements which have the goal to manage digital certificates. By manage we mean to create digital certificates, delete them, check their validity and revoke them.

All this is achieved by building a tree-like structure of Trusted Third Parties (TTP). A TTP is an entity (usually a Certification Authority) trusted by two other entities that allows the first entity to trust the second entity, creating what could be called a “web of trust”. The concept of TTP is one of the key concepts in order to understand how PKI works.

Suppose that an entity A wants to establish a secure connection with another entity, B. With a non secure connection A does not have any guarantee that the connection is really with B, neither can verify the source of the information he is receiving. With PKI, if A and B trust a common Certification Authority they can exchange their certificates issued by this common CA, and as both trust the CA they will trust any entity holding a valid certificate of it.

This can apply also to more complex schemes in which the common CA is various levels up in the PKI hierarchy. On the upper level is the root CA, this CA has an auto-signed certificate and commercial web browsers usually have their certificates pre-installed in them to check their validity.

In order to have a valid certificate, in theory, an entity must ask for it to a CA and prove its identity, then the CA will issue a valid certificate that will be trusted by any entity trusting the CA.

SSL/TLS

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are two cryptographic protocols (SSL being the predecessor of TLS) that provides confidentiality, authenticity and integrity services to the transport layer. These protocols rely typically on X.509 certificates.

MD5

The Message Digest algorithm 5 (MD5) is a cryptographic hash function that generates a 128 bits hash value for a given input message. The message is divided into 523 bits chunks that are processed and compressed. Although it was very popular and widely used nowadays its use is no longer recommended, and it is considered a broken hash function since in 1.995 researchers discovered that is was very easy to generate collisions (two different messages generating an equal hash).

O C S P

The Online Certificate Status Protocol (OCSP) is the protocol employed in PKI to check whether a certificate is revoked or not. The protocol works by querying an appropriate OCSP server asking for the revocation status of a certificate, instead of looking in a CRL, the OCSP checks for the revocation status of the certificate and answers.

Vulnerabilities

This section will show the most relevant vulnerabilities we considered for this small research work. We tried to gather vulnerabilities affecting different aspects of the elements that come to play when making an online purchase.

MD5 harmful today: Creating a rogue CA certificate

This attack was presented in december 2.008, the investigation ran by an international group of researchers (Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger) managed to generate a fake CA certificate checked as valid by the vast majority of programs it was tested on.

This means that the people having this fake certificate could sign and validate any certificate they want and it will be recognised by any browser as a valid certificate. The attack was performed tacking advantage of two critical vulnerabilities:

- **MD5 collisions**: This attack would have been impossible without the vulnerabilities of the MD5 hash function. If the CA uses a hash function to sign that is vulnerable to collisions it means that an attacker can find or generate, under certain circumstances, a certificate that has the same signature, making it valid for any entity trusting the issuer CA.
- **Insecure CA**: The researchers found a CA that met several criteria in order perform the attack. The most critical vulnerability here was the fact that the CA issued the certificates signed with the MD5 function, but they also found several properties that made they work easier due to the automated process to issue certificate that the CA used: the certificates serial number were consecutive. So they could predict which will be the serial number of a new certificate in several conditions, the certificates were issued a fixed number of seconds after they were required.

But even though the CA put things easier and the MD5 properties made the attack possible there was another problem and that was that in order to generate a certificate with the same signature as the one issued by the CA they needed to predict what where going to be the first fields of it (due to the MD5 collision attacks properties). These fields included the issue time and the serial number.

The steps to perform the attack were as follows:

1. Find a CA that issues certificates signed with MD5 hash function. In this case the CA chosen was RapidSSL as its automated process was clearly insecure and had all the properties stated before, and also allowed the reissuing of certificates at a very low cost.

2. After that the researchers studied the statistics of the CA serial numbers generated in its certificates and came to the conclusion in a weekend on average the CA issued around 800-1000 certificates as average. That meant that they could issue a certificate on Thursday and, based on its serial number and time, predict the serial number and time on Sunday night (remember that the CA issued certificates every 6 seconds and have consecutive serial numbers) that the fake certificate will have and also the issued certificate. (The weekend was chosen to perform the issuing because it had less generations and was more predictable).

3. When they had the date and serial number predicted they generated the fake certificate, which could take up to 3 days with a cluster of 200 Playstation 3. This kind of generation means inserting several collision bytes into the message in order to generate the desired hash, but in this case the researcher cleverly hided these bytes into the RSA key making them look random.

4. On Sunday night they issued a certificate before the time predicted and based on the serial number started generating certificates to reach the time predicted with the serial number predicted. This took them "only" 4 weekends, which means only 4 tries and its a really short time considering what they were doing.

5. Once they had a valid certificate and a fake certificate with the same hash, that meant that the signature will be the same for both so they only had to attach the signature of the valid certificate to the fake certificate, making it perfectly valid for any entity trusting the CA.

6. At this point they had a CA certificate that would be taken as valid by any entity trusting RapidSSL (subsidiary of Verisign, which means being trusted by a lot of entities). So they could sign any certificate and it would be checked as valid.

Another characteristic of the certificate generated in this attack is that it is quite difficult to revoke since the common process to check the revocation of a certificate relies on a url provided by the own certificate pointing to a revocation server. Since the rogue certificate had no room to include this url most browsers with their most common configuration will not be able to check the revocation of the certificate.

The "practical" use of this vulnerability is not only issuing valid-looking certificates in order to deploy fraudulent websites with certificates that will be considered valid. This can be also used in conjunction with the famous "DNS flaw" in order to perform more sophisticated phishing attacks that would redirect a user to a fraudulent website and this site would be seen by the user and by the web browser as perfectly legal.

Another important attack that is possible with this rogue certification authority aims to revoke valid and legitimate certificates. Using the method showed in this work an attacker could get a signed certificate with the serial number of the certificate that he/she wants to get revoked after publishing this fake certificate the CA will revoke both certificates with the same serial number since for the revocation process the CA takes into account only the serial number of the certificate, this attack can be very dangerous if its target is the root certificate of the CA since it will force the revocation of the CA root certificate (and also the fake one) along with all the certification tree depending on that root certificate.

This attack was the first real-life implementation of an exploit of the MD5 collision vulnerability. It is a shame that years after the discovery of this vulnerability CAs were still using it to sign certificates. The problem discovered by these researchers is not "only" the fact that CAs where issuing certificates with an obsolete hash function (as it was planned to stop issuing certificates with it in a short period of time), the problem is that nowadays there are a lot of certificates signed with that hash function and the problem is, as stated by the researchers: "What to do with those certificates signed with MD5?".

The answer is very complex as the number of certificates and systems relying on this function is considerable so revoking all the certificates will lead to a chaotic situation as is still widely used, but the alternative to keep using it is not also very comforting as the vulnerabilities of the MD5 function makes it a perfect candidate for any attack.

As a result of this work RapidSSL stopped issuing MD5 signed certificates but did not want to revoke the existing MD5 certificates issued, this may be related with the problems we stated before related to the huge problem that will be generated if all the MD5 signed certificates. We did not found specific information about RapidSSL solving the issuing process vulnerabilities (Consecutive serial numbers, constant time between issuing and request, etc) if not solved those failures could be used again if another vulnerability is discovered, and that cannot be discarded for instance for certificates using SHA-1, then the same problem will appear again.

This research does not stress too much on the problem that these vulnerabilities mean since the main problem is the MD5 collisions, but the vulnerabilities of the CA were the "facilitator" to jump from a theoretical attack to a real attack and may have helped another unknown attacks before. Also, these CA vulnerabilities can be used, if found in a adequate CA, not only in secure SSL communications but also email signing, software signing, etc. and as we have seen previously even can be used to take down an entire certification authority certification tree.

Solving some of the issues exposed in this work will make harder and in some cases almost impossible to perform this attack. Some of the solutions proposed start by discontinuing the use of MD5, since this is quite hard as explained at least stop using it for new certificates is the minimum that can be asked, but here lies another problem since one common substitute for MD5 is SHA-1 and this function has been proved to have similar vulnerabilities so even that nowadays the attacks proposed have high complexity this can be changed soon making this function as harmful as MD5.

In conclusion we think that a key solution for this problem can be to make all the CA to issue all their new certificates using SHA-2 and start a research to begin the substitution of the current MD5 certificates (and even SHA-1) and change them for SHA-2.

Null prefix attacks against SSL/TLS certificates

This attack was the result of the investigation of a celebrity in the scene (Moxie Marlinspike) was made public at the Black Hat 2009 in Las Vegas. In this case the vulnerability used to perform the attack was not related with the hash functions of the certificates or the CAs characteristics, but the attacker is taking advantage directly from the characteristics of the X.509 certificates and the wrong implementation of the validation process.

These vulnerabilities make possible to perform a man in the middle (MiM) attack as we will see in the following section. The vulnerabilities that support the MiM attack are.

1. First of all X.509 certificates define their strings as variations of PASCAL strings, this means that instead of having a set of bytes finished with the null character (\0), the strings in a X.509 certificate will have first a length and then the content of the string, so we could include the null character without any problem in any string of a X.509 certificate.
2. When asking for a certificate, Certification Authorities check the validity of the request by looking at the root domain of the "common name" field of it (being a web address), and sends a confirmation email to the administrator of the root domain. But what the CA does not check is the previous prefix, that means that someone could ask for a certificate for "www.gmail.mydomain.com" and if we are the owners of "my domain.com" we could confirm the request and get that certificate. This characteristic, combined with the previous one leads to the possibility of requesting a signed certificate with a common name as follows "www.gmail.com\0.mydomain.com" and obtaining it with no problem.
3. Most implementations of SSL/TLS operate with the certificate strings as if they were C strings, meaning they end reading a string when they find a '\0', meaning that in our case one of these implementations would think the common name of the certificate is "gmail.com", using a perfectly valid certificate with no manipulations.
4. Worst of all, most of the implementations of SSL/TLS, as its pointed by the author in his research, fall when performing the previous steps, but the Mozilla NSS has a even worse behaviour that accepts wildcards (*) in the common name and makes them match any domain. This means that we will not need to issue a specific certificate for each en-

tity we want to impersonate, we just can use the following url for the common name when requesting the certificate: "*\0.mydomain.com" this will make our certificate able to impersonate ANY domain in any system using NSS (Firefox, Thunderbird, AIM, etc).

Taking advantage of these vulnerabilities, an attacker could intercept a connection from the client side, give the client a certificate that impersonates the website he is trying to access and sign it. On the server side the attacker only would have to establish a regular secure connection with the server and transfer the data between the server and the client.

This scheme will allow any attacker to have unrestricted access to any data that the client sends, and the client will think he is sending the data in a secure way to the server. One important point that can make this attack easier is that most of the web traffic is non-secure . When the connection switches to SSL, it is usually in most of the cases a result of a redirection from an http url to a https or the result of a user clicking on a https link. The beginning of an attack using this vulnerability could be sniffing the traffic of the victim, and when his/her connection switched to SLL and the destination url is a page for which we have a fake certificate we can impersonate the webpage.

This attack relies mainly in the "characteristic" found in the X.509 standard, but this discovery cannot be really surprising since the X.509 standard is, as the author of this work comments, a "complete nightmare" with only 3 revisions in its 20 years of life and parts of it have been literally lost to be found again later. This left us speechless, not much can be said about the X.509 standard apart from the following quote from Robert Jueneman:

"There is nothing in any of these standards that would prevent me from including a 1 gigabit MPEG movie of me playing with my cat as one of the RDN components of the DistinguishedName in my certificate."

A fast and proper countermeasure for this attack is updating the software relying on SSL with this vulnerability (most of it) to meet properly the definition of the X.509 standard. But also here there might be a problem since lot of new software rely on automatic updates, in the case of Firefox an attacker could hijack the connection to the update server and make the program install updates or extensions the attacked may want, this is because Firefox relies in the security of its connection to the update server and installs anything that is presented by the update server with a higher version. Also a revision of the X.509 standard must be take into account since it has lots of problems found and its complexity and "peculiarities" may have led to this vulnerability.

Some implementations like those from Safari or Opera web browsers remove the null character from the Common Name before comparing, this can make us think that they are safe but they are still vulnerable if getting a different certificate, the only difficulty here is that the certificate must be obtained from a CA vulnerable to the null character attack. If we register the domain "secure.ba" we could request a certificate (to specific CAs with the vulnerability) for "secure.ba\oncosantander.es" the CA will validate the certificate for the domain "secure.ba" but the whole string will appear in the certificate, when presented with this certificate, implementations that remove the null character will validate that certificate for "secure.bancosantander.es", making them as insecure as the others.

Even if the attacker success in this attack he/she will have to face another line of defence in the system which is the OCSP which allows the client side to check the revocation status of the certificate and may help the client side to detect the fake certificate if it has been revoked. But this does not means that we are safe as we will see in the following section.

Defeating OCSP with the character '3'

This attack was also researched by Moxie Marlinspike and released at the same time as the previous one. Here also the objective is to perform a MiM attack. But in this case the attacker will be taking advantage from a vulnerability found on the OCSP protocol.

When an entity gets a new certificate it might want to check whether its valid or not in real-time, in that case the entity must make a OCSP request for the certificate it wants to check. The vulnerability found lays in the response that the OCSP sends back with the status of the certificate. If we have faked a certificate like in the previous section and have control over the connection of the client we can intercept the request to the OCSP and fake one that will not compromise our fraudulent certificate in case it has been detected and revoked previously.

The response from a OCSP query has a signature, which at this point we cannot forge, but the signature is only covering the fields in a specific part of the response (ResponseData) which, as points Marlinspike in his work, its optional. These fields will only be used if response validates the certificate it has been checked, so it might seem that there is no way to fake a reply that allow us to use a fake or not valid certificate. But there is, as most implementations of the OCSP protocol have a vulnerability in the processing of the replies from the OCSP server. When the reply status is "try Later" (numerical code 3) most implementations will keep on going without any notification or error message and using the certificate. This reply can be easily faked as it has no "ResponseData" fields thus no data to sign.

To solve this vulnerability software implementations must be updated to have a default action when the "tryLater" response is received. To keep the usability of programs somebody could think that stopping the connection only because of this is not the best idea, but at least some pop-up informing the client about the status and problems faced in this case can be useful.

The other approach to solve this vulnerability is to update the definitions of the protocol so it includes for the signature of the response more fields, specifically the mandatory ones, along with those included already. Although this approach can be the most cost-intensive since changing a widely used standard can be almost impossible, we think it should be performed.

This vulnerability is a good complement to the previous one in order to intercept a secure connection and invalidate all the security checks that can be perform to detect the fraud. If an attacker success with this kind of attack he or she can even prevent the software from being updated in order to correct these vulnerabilities, giving the attacker a great opportunity and leaving the victim completely defenceless.

Other vulnerabilities

While searching for more information we ended up reading some interesting works that we did not want to highlight as much as the previous but deserve a mention, like the research on SSL vulnerabilities in mobile devices (reference 9) based on the vulnerabilities exposed in this work or the more relevant attacks to Extended Validation SSL work (reference 8).

People (final conclusion)

A lot has been said about this vulnerability, even without all the vulnerabilities mentioned before (and more others that we did not cover in our research). As we saw in the previous vulnerabilities they all had at least one big mistake back, whether it is a wrong policy for the certificate issue, a big mistake in the protocol definition or a bad implementation to simplify things.

But even though these errors were corrected we still have another big vulnerability in any secure system: the users. As is stated in the MD5 vulnerability paper, in class and other works we read for this research work, today users are encourage to check if they are on a secure connection by looking for a “padlock” symbol, but that leads us to the question: “What is a secure connection?”.

In all the vulnerabilities that we exposed in this research work the browsers would show a padlock symbol and actually the connection indeed is secure between the holder of the false certificate and the client.

This makes us think that more effort has to be done in these fields: improve and update the definition of the protocols (specially X.509 since it clearly needs to be improved and taken more seriously as we saw before), improving the implementations of the web browsers, mail clients, etc. in order to keep informed the user (expert or non expert) in a more precise way about whats happening.

Also a pedagogic effort must be done in order to inform the users about the risks of relying only on things like the padlock symbol and making them understand the risks beyond the PKI protocol use and implementation of the other protocols seen in this work. As they are mostly secure protocols and work pretty well, failures in specific implementations, in the definition of the standards or in the company policies of a specific enterprise can lead to a catastrophic failure that can even compromise the security of all the system.

As personal conclusions we want to say that we were amazed on the works and examples we read while researching for this work. Its amazing the time and work some people devote to these issues and how they end up publishing it to make things safer instead of taking profit of it (at least that we know of).

References

1. Secure e-commerce course: Class slides and notes.
2. MD5 considered harmful today: Creating a rogue CA certificate. (Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik and Benne de Weger). http://www.packetstormsecurity.org/papers/attack/md5-considered-harmful.pdf
3. RapidSSL Knowledge Base: https://knowledge.rapidssl.com/support/ssl-certificate-support/index?page=content&id=AR1049&actp=search&viewlocale=en_US&searchid=1258720387169
4. US-CERT Vulnerability Note VU#836068
5. Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities (Marc Stevens1, Arjen Lenstra2, and Benne de Weger) http://www.win.tue.nl/hashclash/EC07v2.0.pdf
6. Null prefix attacks against SSL/TLS certificates (Moxie Marlinspike). http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-PAPER1.pdf
7. Defeating OCSP with the character '3', (Moxie Marlinspike). http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatOCSP-PAPER2.pdf
8. Sub-Prime PKI: Attacking Extended Validation SSL, (Michael Zusman and Alexander Sotirov) http://www.blackhat.com/presentations/bh-usa-09/SOTIROV/BHUSA09-Sotirov-AttackExtSSL-PAPER.pdf
9. Tricks for Defeating SSL: effectiveness test on mobile phones, http://www.mseclab.com/?p=180