TLS Connection Validation by Web Browsers: Why do Web Browsers still not agree?

A.S. Wazan[1], R. Laborde[1], D.W. Chadwick[2], F. Barrere[1], *And A. Benzekri[1]*
[1] IRIT Laboratory, Paul Sabatier University
{ahmad-samer.wazan, laborde, barrere, benzekri}@irit.fr
[2]University of Kent
d.w.chadwick@kent.ac.uk

***Abstract*—The TLS protocol is the primary technology used for securing web transactions. It is based on X.509 certificates that are used for binding the identity of web servers' owners to their public keys. Web browsers perform the validation of X.509 certificates on behalf of web users. Our previous research in 2009 showed that the validation process of web browsers is inconsistent and flawed. We showed how this situation might have a negative impact on web users. From 2009 until now, many new X.509 related standards have been created or updated. In this paper, we performed an increased set of experiments over our 2009 study in order to highlight the improvements and/or regressions in web browsers' behaviours.**

Keywords—X.509 Certificate; Certificate Validation; Web browsers

I. INTRODUCTION

Transport Layer Security (TLS) is the primary technology used to secure web communications. Before setting up a TLS connection, web browsers have to validate the TLS certificate of the web server in order to ensure that users are accessing the expected web site. When the web browser displays a small padlock, the user can continue his/her navigation of the distant web site with the knowledge that the web site is who it says it is (if the user pays attention to this [1,19]). However, if the web browser detects a problem with the certificate, a warning message informs the user about this, and the necessity to stop the transaction immediately. In this way, the web browser protects its users from potentially harmful web sites.

Theoretically then, things are relatively simple. But in practice, things are much more complicated than this. For example, on 15 December 2016, around 8am, one of the authors connected to the website of COMPSAC 2017 in order to submit this paper. However, the connection was not possible because the web browser (Firefox) detected an error when validating the website's certificate. Figure 1 is the screenshot of the error message (in English this is "An error happened when connecting to compsac.info. The Online Certificate Status Protocol (OCSP) response contains out-dated information"). It was not possible to submit the paper since Firefox blocked the access without any obvious way to bypass its protection system. Immediately, the author tried to connect to the same submission website using Safari. This time, no error message appeared (Figure 2) and the paper was successfully submitted. A second test with Firefox was then performed but the Figure 1 error message was still displayed. Was the paper securely submitted to the correct website or not?

In 2009, we highlighted the different behaviours of several web browsers (Internet Explorer (IE), Opera and Firefox) when validating certificates [2]. We explained the reasons for these differences were either due to violation of the standards by the browsers, or ambiguity in the standards themselves.

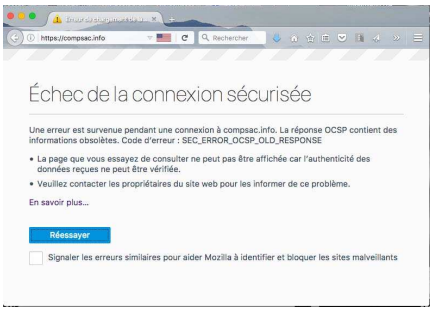

Fig. 1. Screenshot when opening the COMPSAC 2017 submission web site using Firefox

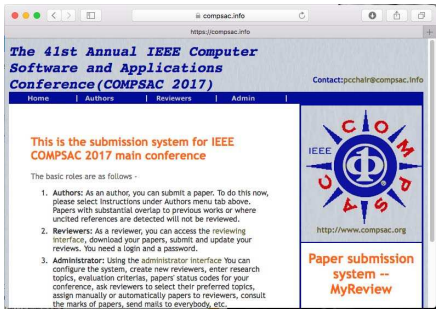

Fig. 2. Screenshot when opening the COMPSAC 2017 submission website using Safari

In this paper, we have performed an increased set of tests from [2] and this time we have covered a greater number of web browsers (IE, Edge, Opera, Firefox, Safari and Chrome), as well as covering the newest standards. Our work describes the quality of X.509 certificate validation implemented by these web browsers, as well as showing their evolution since 2009. Also, we have produced new tests for analysing how web browsers implement the OCSP protocol.

The rest of this paper is structured as follows. Section 2 overviews the base set of standards related to X.509 certificates. Section 3 exposes and analyses the results of tests executed on six web browsers and describes why their behaviours are inconsistent. Finally, in section 4 we conclude.

II. STANDARDS RELATED TO X.509 CERTIFICATES

The contents and processing of X.509 public key certificates (PKCs) are regulated through numerous standards documents. They were first officially described in the X.509 standard developed jointly by the ISO and ITU-T [3]. X.509 provides the general framework for public key infrastructures. This document defines the syntax of PKCs and revocation lists, as well as how they can be extended (by literally anyone). Each standard certificate field has its own syntax and semantics as well as constraints on its possible values. In many cases a field can have different syntax choices. These fields provide information about the certificate version number, the subject of the certificate, the public key, the way the key can be used, and the certificate life cycle management process (Figure 3).

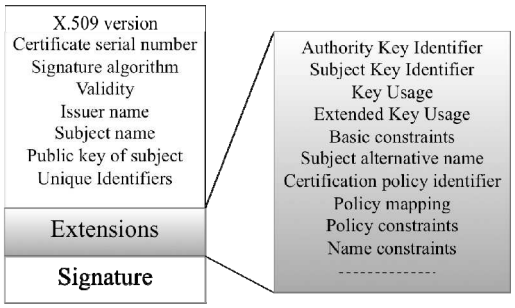

Fig. 3. Certificate contents (inspired by [20])

Three kinds of field exist: mandatory fields, optional fields and extensions (which are all optional). When a field is mandatory, Certificate Authorities (CAs) must fill it and Relying Parties (RPs) must check it when validating certificates. Extensions can be marked as critical or not. If present and marked critical, the RP must obey its contents or reject the certificate. If marked not critical, the RP can ignore the extension if it does not recognise it, but must obey it otherwise i.e. it should not ignore a non-critical extension that it supports.

The complexity of the X.509 standard, in terms of fields that are mandatory, optional, choices, and extensions, means that it is almost impossible for two different implementers to produce interworking code. A PKC produced by one implementer cannot always be fully validated by another, and vice versa.

Consequently the IETF PKIX group developed an X.509 standard profile (RFC 5280) to address the specific needs for using PKIs on the Internet. Especially, the profile eliminates most options, make choices where several are available, and specifies which extensions should be used. However, due to the large constituency of the IETF, many different authors proposed many different extensions and ways of using X.509 certificates, so that by now, over 50 PKIX specified RFCs exist. One can easily see why it is still not a trivial task to implement a fully conformant web browser.

Among all the certificate extensions defined in X.509 and RFC 5280, Internet applications (such as web browsers) must at least be able to recognize: basic constraints, certificate policies, policy constraints, subject alternative name, key usage, name constraints, extended key usage and inhibit any-policy extensions; but do not need to recognize: authority and subject key identifiers, and policy mapping extensions [4].

Other important standards related to X.509 PKCs are:

- RFC 6125: this explains the rules that must be followed in representing and verifying the identity of servers identified in the PKCs,
- RFC 6960: this specifies the OCSP protocol used for checking a PKC's status.
- RFC 5019: this addresses the scalability issues related to the deployment of OCSP servers in high-volume environments. It also specifies the rules to follow for caching OCSP responses.

Other standards will be mentioned in the rest of the paper at the appropriate point.

III. ANALYSIS OF WEB BROWSERS' BEHAVIOR

In 2009, we tested three web browsers: IE 7, Firefox 3 and Opera 9.5. In the current research, we tested the latest versions: IE 11, Firefox 50 (FF50) and Opera 42 (OP42). This allows us to analyse the evolution of their PKC validation processes. We also evaluated three new major web browsers: Microsoft Edge 38 (ED38), Google Chrome 55 (GC55) and Safari 10 (SA10). These are preceded with an asterisk (*) in the table of results to highlight that they were not included in the 2009 study.

Since our goal is to understand the exact certificate validation processes performed by web browsers when initiating TLS secured communications, we tested their responses when they were confronted with chosen test values in specific certificate fields. The results were then analysed and compared to the expected behaviour described in the relative standards.

Because we want to examine the evolution of PKC validation practices, we performed the same tests as in 2009 and added new test cases about the subject field and key usage extension. These new tests are preceded with an asterisk (*) in the table of results.

When handling TLS certificates, web browsers return one of three possible responses, denoted as follows:

- A: accept the certificate without any intervention by the user,
- W: inform the user about the existence of a problem by showing a warning message and asking him/her to take a decision,
- R: refuse the certificate and prohibit access to the web server without any intervention by the user.

To easily identify the evolution of web browsers' behaviour compared to the study of 2009, we use the symbol ↘ whenever the evolution is considered a regression and the symbol ↗ whenever the change is an improvement. In addition, we highlight the results that are not conformant to standards by colouring the cells in grey.

Our technical test environment is as follows: OP42, GC55, FF50 and SA10 are deployed on Mac OSX Sierra; IE11, ED38 are deployed on Windows 10; our Web server is Apache/2.4.18 and is installed on an Ubuntu Server 16.04.1 LTS. We generated all the test certificates using OpenSSL 1.0.2.

A. TLS Certificate Subject

The TLS certificate subject represents the identity of the web server. This may be either a Fully Qualified Domain Name (FQDN) or an IP address or both. FQDNs and IP addresses are different types of name (called name forms in the standards). A web server could hold many FQDNs that all point to the same IP address, e.g. as in virtual hosting. Conversely, one FQDN may point to different IP addresses (e.g. for load balancing).

1) What do the standards state about the subject?

The X.509 standard [3] states that the subject field identifies the entity associated with the public-key found in the subject public key field. An entity could have one or more alternative names, of different types (or forms), held in the *subjectAltName* extension. According to the X.509 standard, an implementation that supports this extension is not required to process all the name types. If the extension is flagged critical, at least one of the name types that is present must be recognized and processed, otherwise the certificate must be considered invalid.

RFC 5280 [4] states that the subject name may be carried in the subject field and/or the *subjectAltName* extension. If the subject naming information is present only in the *subjectAltName* extension, then the subject name should be empty and the *subjectAltName* extension must be marked critical. According to this statement a TLS certificate can hold multiple names in a combination of the Subject field (CN component) and the *SubjectAltName* extension. These names must all refer to the same entity, although a browser need not recognize all the different name types.

2) Test and Results

We performed two types of experiments to test certificates holding the FQDN and IP names separately, as well as both types together.

In the first set of experiments, we tested how the web browsers reacted when the certificate contains zero, one or more FQDN names. We configured our web server to respond to requests sent to either www.server1.com (S1) or www.server2.com (S2). As the names could be mentioned in either or both of the Subject Name-Common Name (SCN) and *SubjectAltName*-DNS Name (SAN-DNS) fields, we tested the following different combinations of names in our web server certificate:

1. SCN=www.server1.com, SAN-DNS=www.server2.com
2. SCN=null, SAN-DNS=www.server2.com
3. SCN=www.server1.com, no SAN-DNS field
4. SCN=null, no SAN-DNS field
5. SCN=null, SAN-DNS=www.server1.com and www.server2.com.

For each combination, we recorded the reaction of each web browser when accessing www.server1.com and www.server2.com (Table I). We also state whether the certificate is Valid (V) or Invalid (I) according to the X.509 standard, RFC 5280 [4] and RFC 6125 [5]. Because we obtained the same results when the *SubjectAltName* extension was marked critical or not, we haven't indicated this in Table I. The expected results would be that Valid PKCs are Accepted, and Invalid PKCs are either Refused or a Warning given. All browsers behaved as expected.

In the second set of experiments, we tested how the browsers reacted when accessing either www.server1.com or IP address 192.168.57.2 when: i) an IP address only, or ii) iii) an IP address and a FQDN, or iv) a FDQN only, or v) neither, are used in the PKC to identify the web server. In all cases except ii) the SCN field was null. With regard to the study of 2009, test ii) is a new test case in which the SCN field is set to www.server1.com and its IP address is set in the IP component of the *subjectAltName*. All browsers behaved as expected, except for test ii) (see below). We obtained the same results when the *subjectAltName* was marked critical or not, so we have not shown these results in Table II.

3) Analysis of the Results

The primary objective of an X.509 PKC is to bind an identity to a public key. In the case of a web server, the identity is either a FQDN name or an IP address. When the identity of the server is null (Table I iv) Table II v)) the browser cannot authenticate the server, so the TLS certificate is invalid. Whether a browser should immediately refuse an invalid certificate (R) or ask the user what to do (W) is partly a usability issue and partly a security issue. But it is not a standard's issue. The standards will only give guidance on whether a certificate is invalid or not, but will not advise a RP what to do with it. From a security perspective, if the browser (the RP) cannot authenticate the web server, the certificate should be rejected (R). From a usability perspective the user could be given a choice (W), although in practice most users simply click OK to all the pop up windows so invalid certificates end up being accepted. RFC 5280 mandates that the IP address if present must contain either four (for IPv4) or sixteen (for IPv6) octets, and that the FQDN if present must not be null. So the Table II v) certificate is clearly invalid. But none of the browsers rejected it. Instead they ask the user what to do. It should be noted that the way the web browsers present the warning message is different. Some warnings are more difficult to ignore than others. In the case of Safari, ignoring a warning message requires one action (click on "Continue") whereas with GC55 it requires two actions (click on "Advanced" then on "Proceed to …").

If the standards are not clear about a certificate's validity, this can lead to web browser implementers holding different interpretations of this. In the study of 2009, we raised one ambiguity about the validity of a certificate that holds two FQDNs: one in the SCN field and the other in the SAN-DNS extension (Table I i)). The behaviour of the tested web browsers was different. IE7 and FF3 treated the certificate as invalid, whilst OP9 treated it as valid. To cope with this issue, a new RFC (RFC 6125 [5]) was issued in 2011 to handle the ambiguity. The most important recommendations are summarised as follows [5]:

- Move away from including and checking strings that look like domain names in the subject's Common Name.
- Move toward including and checking DNS domain names via the *subjectAlternativeName* extension designed for that purpose: *dNSName*.

TABLE I. MULTIPLE FQDN WEB SERVER IDENTITIES

	IE11		FF50		OP42		*ED38		*GC55		*SA10		Standards	
By address / Values in fields	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
i) SCN=S1, SAN-DNS=S2	W	A	W	A	W↗	A	W	A	W	A	W	A	I↗	V
ii) SCN=Null, SAN-DNS=S2	W	A	W	A	W	A	W	A	W	A	W	A	I	V
iii) SCN=S1, no SAN-DNS	A	W	A	W	A	W	A	W	A	W	A	W	V	I
iv) SCN=Null, no SAN-DNS	W	W	W	W	W	W	W	W	W	W	W	W	I	I
v) SCN=Null, SAN-DNS=S1,S2	A	A	A	A	A	A	A	A	A	A	A	A	V	V

Where: **S1** = www.server1.com, **S2**= www.server2.com, **V**=Valid, **I**=invalid

TABLE II. IP ADDRESS SERVER AND/OR FQDN IDENTITIES

	IE11		FF50		OP42		*ED38		*GC55		*SA10		Standards	
By address / Values in fields	S1	@IP	S1	@IP	S1	@IP	S1	@IP	S1	@IP	S1	@IP	S1	@IP
i) SAN-IP=192.168.57.2	W	A↗	W	A	W	A↗	W	A	W	A	W	A	I	V
ii)*SCN=S1, SAN-IP=192.168.57.2	A	A	W	A	W	A	A	A	W	A	A	A	?	V
iii) SAN-DNS =S1, SAN-IP=192.168.57.2	A	A	A	A	A	A	A	A	A	A	A	A	V	V
iv) SAN-DNS =S1, no SAN-IP	A	W	A	W	A	W	A	W	A	W	A	W	V	I
v) SAN-DNS =Null, no SAN-IP	W	W	W	W	W	W	W	W	W	W	W	W	I	I
vi) SAN-DNS =Null, SAN-IP=192.168.57.2	W	A	W	A	W	A	W	A	W	A	W	A	I	V

Where: **S1** = www.server1.com, **@IP**=192.168.57.2, **V**=Valid, **I**=invalid

However, this RFC doesn't invalidate completely the setting of DNS names in the SCN field as it states: "*In general, this specification recommends and prefers use of subjectAltName entries (DNS-ID, SRV-ID, URI-ID, etc.) over use of the subject field (CN-ID) where possible….. However, specifications that reuse this one can legitimately encourage continued support for the CN-ID [SCN] identifier type if they have good reasons to do so, such as backward compatibility with deployed infrastructure*"

According to the same RFC, if the DNS name is set in both the SCN field and the SAN-DNS, the certificate must be treated as invalid if the user tries to access a web server using the DNS name present in the SCN field: "*A client MUST NOT seek a match for a reference identifier of CN-ID if the presented identifiers include a DNS-ID, SRV-ID, URI-ID, or any application-specific identifier types supported by the client*".

Thanks to this clarification of the standards, the behaviour of web browsers has now become conformant. All of them show a warning message to web users whenever they try to access a web server using the DNS name contained in the SCN field and in the presence of a different DNS name in the SAN-DNS field.

[4] says that web browsers must "recognize" the SAN extension, but only that "*all parts of the subject alternative name MUST be verified by the CA*". This does not place any requirements on the web browser to do likewise. Similarly [3] states "*An implementation is not required to be able to process all name forms*". So browsers do not have to support SAN-IP.

In 2009, IE7 and OP9 didn't support SAN-IP, so they didn't recognise the IP name of the server. FF3 on the other hand did support the IP name form and so did recognise the server's name. Today, however, all the tested web browsers support the SAN-IP. All of them accept a PKC with the correct SAN-IP whenever the web user accesses the web server by its IP address. However, we have found a new ambiguity in the standards about the validity of a certificate that holds a DNS name in the SCN field and a matching IP address in the SAN-IP component, and is accessed via its DNS name (Table II ii). X.509 implies that such a certificate is valid, but RFCs 5280 and 6818 are silent about the validity of such a certificate. For this reason, the behaviour of web browsers is different: SA10, IE11 and ED38 consider it Valid and accept it, whereas GC55, FF50 and OP42 consider it Invalid and issue a Warning message. This test case was not tested in 2009.

It should be noted that the test case is legitimate. A web server, which uses the SCN field to hold the DNS name, may want to add its IP address to its PKC. The only suitable place is the IP component of the SAN. Except for this issue, our tests show a general improvement in the behaviour of web browsers as they are accepting (A) valid certificates, and refusing (W or R) invalid certificates.

B. Key usage, extended key usage

Key usage and extended key usage are used to determine the purpose of the public key contained in the PKC. A TLS server certificate could have a key usage extension or not. The standards [4][3] don't constrain the authorities to issue TLS certificates with key usage extensions.

1) What do the standards state about the Key Usage and Extended Key Usage extensions?

The X.509 standard [3] states that if either the extended key usage or key usage extensions are recognized by the Relying Party then the PKC must be used just for the purposes indicated in it. The key usage and the extended key usage must be treated separately but they must have consistent values. If there is no purpose consistent with both fields, then the certificate shall not be used for any purpose [3].

RFC 5280 states that the key usage extension, when it appears, should be a critical extension. For a TLS certificate, RFC 5280 recommends that the key usage, when it is defined, should have the value of "*digital signature, key encipherment and/or key agreement*" and the consistent value of the extended key usage should be "*Server Authentication*".

2) Tests and Results

The value needed in the key usage extension depends on the encryption algorithms used for generating the certificate's keys (RSA, DSA, DH, etc.) and on the cipher suite applied in the TLS communication between the client and the web server. A cipher suite consists of a key exchange scheme, a signature algorithm, a block cipher algorithm, and a hashing algorithm for computing the authentication key. They're usually identified in a string [6]:

[SSL/TLS]_[key exchange]_[signature algorithm]_ WITH_[block cipher]_[authentication hash]

For example, TLS_ECDHE_RSA_WITH_AES256-GCM_SHA384 is a cipher suite that implements Elliptic-curve Diffie-Hellman Ephermal key exchange algorithm and uses the RSA algorithm as the signature algorithm with AES256 Galois/Counter Mode as the block cipher and SHA384 for the authentication hash.

We generated our test certificates using the RSA algorithm. In this case, two types of cipher-suites are possible:

- TLS_ECDHE_RSA*: in this case, the key exchange algorithm is ECDHE. This means that the RSA private key of the server's certificate will be used for signing the ECDHE public key and the associated parameters [7, page 20]. The appropriate value of the key usage extension is *digitalSignature*.
- Or TLS_RSA_*: in this case the key exchange algorithm is RSA. This means that the client will use the RSA public key of the server's certificate for encrypting the random value chosen by the client (pre-master secret). The appropriate value of the key usage extension is *keyEncipherment*.

Since RSA keys can lead to different key usages, we first check the cipher suites agreed between our web server and the web browsers by looking at the Hello server message in the TLS protocol. Table III shows all of them chose ECDHE for exchanging the key. Thus, the appropriate key usage value must be *digitalSignature*.

We tested how the web browsers reacted when they validated a certificate, which conveyed an RSA public key and had a key usage value different from "*digitalSignature*". It should be noted that the same results were obtained when the key usage was critical or not, which is correct. We chose wrong values "*keyAgreement*", "*dataEncipherment*", "*keyEncipherment*" and the correct value "*digitalSignature*" as test values for the key usage extension. The final column of Table IV indicates whether the certificate is valid or invalid according to the standards.

3) Analysis of Results

Here, the diversity of the web browsers' behaviour is due to their inability to detect violations of the standards when the key usage extension contains wrong values. Most invalid PKCs were accepted by all the tested web browsers. However, in one positive case, FF50 rejected the server's certificate when it contained the dataEncipherment value. This behaviour is an improvement over the behaviour of FF3 in 2009. However, FF50 and OP42 still accept invalid PKCs whose key usage extension is keyAgreement (KA) or keyEncipherment (KE). FF3 and OP9 behaved correctly in 2009 and rejected these PKCs. Finally, as in 2009, IE11 accepts certificates when the key usage has wrong values of dataEncryption (DE) or KA instead of Digital Signature (DS). It is quite appealing to observe that Table IV shows almost unanimous non-conformance against the standards. We cannot determine completely whether the non-conformance of the web browsers is due to some compatibility issues or not.

When the extended key usage has the wrong value of client authentication instead of server authentication, all web

TABLE III. CHOSEN CIPHER SUITES

	IE11	FF50	OP42	*ED38	*GC55	*SA10
Cipher suite	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TABLE IV. KEY USAGE TEST

	IE11	FF50	OP42	*ED38	*GC55	*SA10	Standards
KU=KA and EKU absent	A	A↘	A↘	A	A	A	I
KU=DE and EKU absent	A	R↗	A↘	A	A	A	I
KU=KE and EKU absent	A	A	A	A	A	A	I
*KU=DS and EKU absent	A	A	A	A	A	A	V
KU=KA and EKU=SA	A	A↘	A↘	A	A	A	I
KU=DE and EKU=SA	A	R↗	A↘	A	A	A	I
KU=KE and EKU=SA	A	A	A	A	A	A	I
*KU=DS and EKU=SA	A	A	A	A	A	A	V
KU absent and EKU=CA	R	R↗	R	R	R	W	I
*KU=DS and EKU=CA	R	R	R	R	R	W	I

Where: **KU:** Key Usage extension, **EKU:** Extended Key Usage extension **DE:** dataEncipherment, **DS :** DigitalSignature, **KE:** keyEncipherment, **KA:** keyAgreement, **CA:** ClientAuth, **SA:** ServerAuth

browsers reject the PKC. However, SA10 shows a warning message to the user instead of blocking access to the website. We are not convinced that SA10's behaviour is very helpful, since this will invariably result in an invalid certificate being accepted by the user. The behaviour of FF50 is considered an improvement because in 2009 FF3 accepted PKCs with this wrong value of the extended key usage extension.

C. Revocation

The primary objective of revocation is to remove an invalid certificate from circulation as quickly as possible. This is usually done by asking the RP to check the certificate's status before accepting it.

CAs can revoke a PKC by either publishing its serial number in a Certificate Revocation List (CRL) that can be downloaded from a repository, or by running a specialized server that can be accessed by the Online Certificate Status Protocol (OCSP) [8]. *CrlDistributionPoints* (CDP) and *AuthorityInfoAccess* (AIA) extensions are used to hold the CRL and the OCSP indicators respectively in a certificate.

In general, most of the RP agreements state that RPs are responsible for taking the risk of using revoked certificates. As a result, RPs must be aware of the PKC's status before using it in a transaction.

In this study, we developed more advanced tests for the OCSP protocol. For this reason, we start by giving a brief description of it. The OCSP protocol is described in RFC 6960 [8]. As shown in figure 4, when a web browser gets the server's certificate, it retrieves the address of the OCSP server (responder) from the AIA extension. The browser formulates an OCSP request, which contains the ID of the server's certificate. The browser may send the request using either the HTTP GET or POST methods. Upon receipt of the OCSP request, the OCSP server sends a signed response (in DER format) that contains the certificate ID and its status: '*good*', '*revoked*', or *'unknown'*. It also contains *thisUpdate* and *nextUpdate* fields. The former is mandatory, the latter is optional. *thisUpdate* is used to represent the most recent time at which the responder knows the indicated status to have been correct [8]. *nextUpdate* represents the time at or before new information about the PKC's status will be available.

When the web browser receives an OCSP response, it can store it in a local cache for a period that corresponds to the time between the *nextUpdate* and *thisUpdate* fields [9]. Ideally, web browsers must verify the freshness of the OCSP response in order to avoid relying on out-of-date or replayed responses [9]. Two methods can be used to avoid this: adding nonces to the OCSP requests and responses, or verifying the time value in the *thisUpdate* field. In section C.2, we will see how the different web browsers handle the generation of OCSP requests, and the analysis of OCSP responses and their caching.

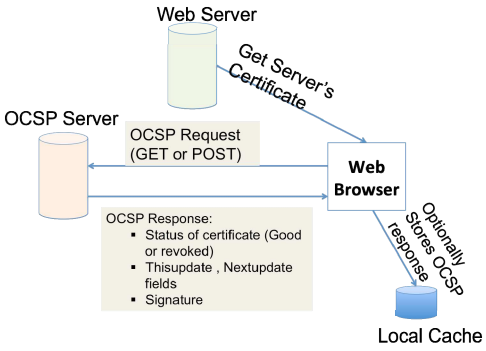

Fig. 4. The OCSP protocol flow

1) What do the standards state about the CRL Distribution Points and Authority Info Access Extensions?

The X.509 standard states that the CDP extension can be, at the option of the certificate issuer, critical or not; but it recommends it to be non-critical for interoperability reasons. When it is a critical extension, a certificate-using system shall not use the PKC without first retrieving and checking the CRL [3]. However, when the extension is not critical a certificate-using system can use the PKC only if the revocation checking is not required by local policy or it is accomplished by other means [3]. According to RFC 5280, the CDP and AIA extensions should be non-critical extensions, but it recommends supporting these extensions by CAs and applications [4].

2) Tests and Results

In the first experiment, we determined the approaches to revocation supported by each web browser, and if it is automatically configured or not (Table V i & ii).

In the second experiment (Table V iii & iv), we show the reaction of web browsers when the OCSP server is down or when the CRL is not retrievable.

In the third experiment (Table V v), we show the reaction of web browsers when they get an OCSP response indicating that the server's certificate is revoked.

In the fourth experiment (Table V vi), we show the HTTP methods supported by the web browsers.

In the fifth experiment (Table V vii), we show which web browsers cache OCSP responses that indicate the PKC is revoked. From this table, we can identify the browsers that use local caches for storing all types of OCSP responses (i.e. not only '*revoked*' type responses).

TABLE V. REVOCATION TESTS

	IE11	**FF50**	**OP42**	***ED38**	***GC55**	***SA10**
i)CRL checking	Automatic, configurable	Not Supported	Not Supported	Automatic	Not Supported	Automatic
ii)OCSP checking	Automatic, configurable	Automatic, configurable	Not Supported	Automatic	Not Supported	Automatic
Where: Automatic means that the browser checks the certificate status automatically. Automatic, configurable means that the browser checks the certificate status automatically, but the user can disable this option.						
iii)OCSP server is down	A/W configurable	A/R configurable	N/A	A	N/A	A
iv)CRL is not retrieved	A/W configurable	N/A	N/A	A	N/A	A
Where : N/A means not applicable. A means Accept. A/W configurable: means that the browser accepts the certificate whose revocation status is not verified, but the user can activate an option to get warned when the certificate status is not verified. A/R configurable similar to A/W configurable but instead of showing a warning, the certificate is refused and the access is blocked.						
v)*Certificate is revoked in the OCSP response	R	R	R_U	R	R_U	W
Where : R means that the browser rejects the certificate after receiving the OCSP response. R_U means that the browser rejects the certificate by reading its status from the cache of the underlying OS system. W means that the browser shows a warning message upon the reception of the OCSP response; the user has the possibility to get into the website.						
vi)*OCSP request HTTP methods	GET, POST, automatic	POST automatic, GET manual	**N/A**	GET,POST automatic	**N/A**	GET automatic
Where : GET, POST, automatic means that the browser uses by default the GET method to retrieve the OCSP response, if the OCSP server doesn't support the GET request, the browser will automatically send a POST request. POST automatic, GET manual means that the browser uses the POST method by default, the user can activate the use of the GET method. However, the browser will not use the POST method if the GET request fails. GET automatic means that the browser supports only the GET method, if it fails the browser will not send a POST request. N/A means not applicable.						
vii)*Revocation response caching	C	DC	**N/A**	C	**N/A**	C
Where :C means that the browser stores the OCSP response that indicates the revocation of a certificate in a cache. DC means don't cache . N/A means not applicable.						
viii)*cashing response when *nextUpdate* field is absent	C	DC	N/A	C	N/A	C
Where :C means that the browser stores the OCSP response in a cache. DC means don't cache . N/A means not applicable.						
ix)*checking freshness of OCSP response	DCF	DCF	N/A	DCF	N/A	DCF
Where :DCF means don't check freshness. N/A means not applicable.						

In the sixth experiment (Table V viii), we show which web browsers will cache OCSP responses whose *nextUpdate* field is absent.

Finally in the last experiment (Table V ix), we show whether web browsers verify the freshness of OCSP responses or not.

3) Analysis of the results

The inconsistency of the revocation processes comes from the different implementation efforts by the web browser manufacturers and not from that of the CAs suppliers.

Maintaining a revocation service (either CRLs or OCSP) is a requirement for CAs. The standards [3][4] also recommend, but do not mandate, that RPs should ensure that the PKCs are not revoked before they rely on them. However, when the AIA and CDP extensions are present and understood, the RPs are required to process them. X.509 states about the CDP extension: "*a certificate-using system shall not use the certificate without first retrieving and checking a CRL from one of the nominated distribution points*" Therefore browsers should not ignore these extensions and they should fetch the revocation information and check it before accepting a certificate.

There is some ambiguity over what should happen when a CA claims it maintains an OCSP service but does not. RFC 6960 [8] states "*the OCSP client suspends acceptance of the certificate in question until the responder provides a response*" and "*In the event that the OCSP responder is operational but unable to return a status for the requested certificate, the "tryLater" response can be used to indicate that the service exists but is temporarily unable to respond*.". In the second experiment (Table V iii), the OCSP server was down and didn't send any "*tryLater*" response. Thus, the reaction provided by the web browsers is not fully conformant as none of them block the access.

Clearly, authorizing the access to a website whose certificate status is not verified compromises the security of web users. In 2009, there were three different browser actions: IE7 authorized the access (*soft-fail*), whilst FF3 blocked the access (*hard-fail*). OP9 authorized the access but removed the pad-lock icon and asked the user not to send sensitive information (*soft hard-fail*). Our tests in this paper show that today all the web browsers apply the principle of *soft-fail*. According to Adam Langley from Google [10], browsers went in this direction because they considered that hard-failing raises a different security issue by creating a single point of failure: if every web browser would hard-fail the connection, then OCSP servers would be the easiest target for DDOS attacks.

Because *soft-fail* is the preferred action, Google has reached the conclusion that dynamic revocation checking is useless [10]. As a result, Google decided in 2012 to stop checking the status of *non-ExtendedValidation (EV)* certificates. Instead, Google invented a new revocation technology called CRLSets [11]. The basic idea of CRLSets is that Google merges all the CRLs of the existing CAs and reduces the obtained list by removing PKCs that it considers unimportant. The result is a minimal CRL list that is periodically pushed to Google Chrome. Opera seems to be following Google's approach. The other web browsers (IE11, ED38, SA10 and FF50) still support OCSP and CRL checking (Table V i & ii).

Curiously, when the Heartbleed bug was disclosed publically in April 2014 [12], millions of TLS certificates had to be revoked in a very short time. This event confirmed the limit of the traditional CRL revocation approach (in one case it was reported that a CRL file grew from 22 KB to 4.7 MB [13]). However, OCSP responders didn't face any serious performance problems as a result of Heartbleed [14]. To cope with this issue, many advanced web users wanted to clean their revocation cache and to activate the hard-fail OCSP option. Google Chrome users were able to enable the revocation checking in their advanced settings but today, in the latest version of Chrome, this option has disappeared! So these users cannot hard-fail their connections because of Google's approach. This shows the limits of CRLSets and highlights the need for a complete revocation checking mechanism.

Whilst the OCSP validation process was not affected by the Heartbleed bug, nevertheless it has a performance overhead on TLS connections, since the PKC's revocation status is checked each time in parallel with establishing the TLS connection. An OCSP exchange may take up to 350 ms per HTTPS connection [15]. To improve OCSP performance web browser manufacturers have implemented several caching approaches.

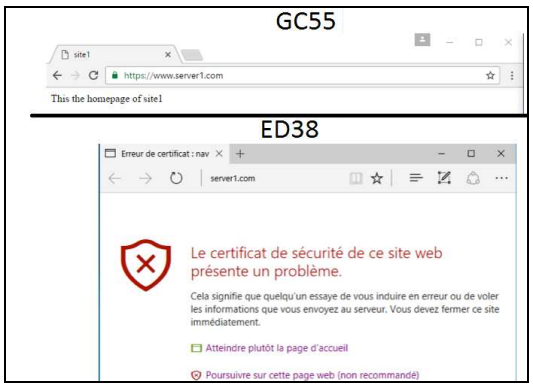

Fig. 5. Comparison of Edge and Chrome for a revoked certificate

Table V v) shows the reaction of web browsers when they are notified that a PKC is revoked, and Table V vii) their approach to caching. Under macOS Sierra, SA10 checks for the status of the PKC and stores the result in the local cache (located at ~/Library/Keychains/*/ocspcache.sqlite3). On Windows 10, there are two different caches. The first one is used by IE11 and the second by ED38. Finally, Table V vii) shows that although GC55 and OP42 don't directly check for the certificate status, they do read the local cache of the underlying OS system in order to inform the web user about the revocation. On Windows 10, GC55 reads the local cache of IE11, not that of ED38. Figure 5 shows how ED38 indicates that a PKC has been revoked while GC55 tells the user that the connection is secure.

The rest of our tests show that the three web browsers (IE11, ED38 and SA10) that support the OSCP mechanism with caching, still don't respect the standards in some respects. For example, a web browser can cache the status checking response for a period between the *thisUpdate* and *nextUpdate* times. When an OCSP server sends an OCSP response without the *nextUpdate* field, the web browser should not cache the status response but must be reject it. RFC 5019 states: "*Clients MUST check for the existence of the nextUpdate field and MUST ensure the current time, expressed in GMT time as described in Section 2.2.4, falls between the thisUpdate and nextUpdate times. If the nextUpdate field is absent, the client MUST reject the response*". However, RFC 6960 states that "*If nextUpdate is not set, the responder is indicating that newer revocation information is available all the time.*". This means that web

browsers can accept the OCSP response but they must not cache it. This inconsistency between the two standards comes from the fact that RFC 5019 is a profile of RFC 6960. RFC 5019 appears more restrictive than RFC 6960 about the presence of the *nextUpdate* field in the OCSP response. However, nothing prevents a web browser to adopt the liberal approach of RFC 6960 rather than the restrictive one of RFC 5019. It should be noted that in all cases the web browsers must not cache this type of OCSP response. Our test shows that IE11, ED38 and SA10 have stored the OCSP responses for a period up to 24h (Table V viii).

Similarly, Table V ix) shows that these web browsers, as well as FF50, either don't check the freshness of the OCSP responses, or assume that less than 1 day is fresh, when the *nextUpdate* field is absent. To prove this, we stored an OCSP response that indicates the certificate status as "Good" and later we revoked the certificate. Instead of sending the real response that comes from the OCSP server, we sent our stored OCSP response to the web browsers. IE11, ED38, FF50 and SA10 accepted our stale response for a period of up to 24h. It should be noted, that the hard-fail option of FF50 was set to true. The fact that FF50 didn't show any error message means that it has accepted the OCSP response that we sent. For the other web browsers, the fact they cached our OCSP response means that they accepted it.

RFC 6960 [8] requires web browsers to send OCSP requests using either the GET or POST methods. All the concerned web browsers respect this issue. However some OCSP servers may support only one OCSP request method (POST or GET), e.g. our OCSP test server only supported the POST method. Our tests show (Table V vi) that only IE11 and ED38 support the two methods. RFC 6960 [8] should clarify this issue.

Recently, all the web browsers have introduced an additional approach for checking a PKC's status, called OCSP stapling [16-18]. In OCSP stapling, CAs issue certificates with this new extension, which requires the web server to send a cached OCSP response in the TLS handshake. Web browsers should ensure this stapled OCSP response is present otherwise they should *hard-fail* the TLS connection. This approach offers three main advantages. First, it reduces the costs for the CAs because the number of OSCP request is significantly reduced, coming only from web sites. Secondly, it improves the privacy of web users because CAs can't know the web sites users are visiting. Thirdly, it improves the performance of web browsers, as a second connection to an OCSP server does not need to be established. However, this solution doesn't resolve the problem of a single point of failure and DDOS attacks mentioned by Google. An attacker can still attack the OCSP servers of a CA to prevent web servers from fetching new OCSP responses. As a consequence, access to these web sites would still be blocked if *Must Staple* is activated. This is why Google is continuing to not implement OCSP or the must-Staple option in Chrome. Netcraft state [14] that removing OSCP checking from Chrome provides Google with a performance advantage over the other web browsers like FF50 that do support it, but it is at the cost of leaving users in the dark about revoked non-EV certificates.

From the web user's point view, the current revocation practices are completely obscure. Many studies show that most web users don't understand either what a PKC is or the role of a CA [1]. For those who do understand these things, they still cannot easily explain what is happening in some cases. For example, in order to understand the differences in the behaviour of SA10 and FF50 concerning the Compsac2017 revocation message, as described in the introduction, we had to setup a complex test environment and spend several hours understanding the finer details of the OCSP standard in order to explain the origin of the OCSP error message, which was *OCSP_OLD_response*. Our analysis is the following: the *OCSP_OLD_response* error message can be generated because of a problem with the local clock of either the web user (possibility1) or of the OCSP server (possibility2), or it can come from an attacker who tries to send an out of date response (i.e. replay attack) (possibility3). After some investigation, it turned out that our machine was connected to a network time server of Apple, and for some reason, the access to an Apple Time server was blocked by our network administrator. The reason why FF50 refused the connection and SA10 accepted it, is that FF50 had been configured to Refuse whilst SA10 automatically Accepts bad OCSP responses (see Table V iii)).

We believe that the revocation process of web browsers should be improved by: (1) Helping users to clearly see and have more control over the revocation process regardless of the approach used (OCSP, CRL, etc). Specifically, users should be able to change the process according to the context of use. E.g. if a web user connects to his/her bank, (s)he may need to activate the hard-fail option and obtain the freshest verification; but if (s)he connects to a TV server, (s)he might tolerate a soft-fail option and a stale verification. Today, none of the web browsers consider such a need. (2) Giving users the ability to easily flush their OCSP/CRL caches. E.g, on 13 October 2016, GlobalSign wrongly revoked an intermediate CA certificate [21]. Many web users had this bad response in their caches and as a result, access to different web sites was suddenly blocked. Globalsign issued guidelines to help web users clean their caches in order to obtain a fresh response, but this guide was not complete - it didn't include ED38.

IV. Conclusion

Our tests show that considerable changes in the validation process of web browsers have been observed. However, their behaviour is still inconsistent. Some behaviour is dangerous for web users. For example, our test cases show that SA10 always shows a warning message regardless of the seriousness of the certificate validation error. Even when a server's certificate is revoked, the web user has the possibility of proceeding to the web site.

Clearly, the inconsistency of web browsers regarding PKC validation confuses web users. The reasons behind the inconsistent behaviour are: (1) Standards are complex, vague and allow different implementations for different contexts of use. (2) There are a multitude of standards (~50) that handle validation issues. (3) There appears to be no real coordination between the web browser suppliers regarding the PKC validation process, specifically:

a. Web browsers have different styles of validation warning messages.

b. Web browsers have different trust admission policies for CAs.

c. Web browsers handle the ambiguity in the standards differently.

d. Web browsers offer users different preferences/settings related to the validation process. For example, users of FF50 have the choice to de/activate the OCSP hard fail, OCSP stapling, and OCSP must stable options. The same options are not present on SA10 and IE11.

We believe that web browser implementers still have some way to go before their implementations consistently implement the X.509 standards in a user friendly way. Disambiguating standards or introducing another standard won't solve these issues. As mentioned before, the standards give only guidance on whether a certificate is invalid or not, but will not advise an RP what to do with it. Having an advisory authority for the browser industry would help to improve the consistency of PKC validation by web browsers. We believe that the CAB-forum is the right place to handle PKC validation issues. Today, the main focus of the CAB-forum is the issuance of certificates rather than their validation. Extending the scope of the CAB-forum to consider validation issues would have a positive impact on web users. Ideally consistency should not only be applied to browser certificate validation, but also to browser configuration, PKI error messages and processes and procedures. Through the CAB-forum, web browser suppliers could coordinate more effectively and avoid taking individual initiatives. For instance, the decision made by Google to abandon the verification of certificates' status in Chrome without any coordination with the other web browser suppliers has only served to increase confusion in the Web TLS system. Paradoxically, Google has launched a new PKI trust service [22] in which one of the user's/RPs' obligations is: "*(c)* ***checking Certificate status****, and the validity of all Certificates in the applicable Certificate's chain, before you rely on a given Certificate*". If only Google would apply this obligation to Chrome!

Finally, we believe that Web browsers must assume the responsibility of their actions vis-à-vis RPs. Since the creation of the SSL protocol, the exact role of a web browser in the validation process has never been clearly defined. The Web PKI industry has defined the obligations of PKIs, certificate holders (i.e. web servers' owners) and RPs (meaning web users), but not of web browsers. Their role in PKC validation is very important because they are the entities that control the whole validation process on behalf of the users. Specifically, they are the entities that include the trusted CAs in the users' platforms, they validate the certificate fields on users' behalf, but they are the only entities that have nothing to lose if revoked certificates are accepted. What will happen if a web user is connected automatically by a web browser to a fraudulent website whose certificate has been revoked? The web browser vendor's response will be to admit no liability. The web user may lose his/her sensitive information, or download malware, and at the same time, (s)he cannot seek compensation from the relevant PKI nor the web domain owner because the former has respected its obligations to indicate the revocation of the certificate, and the latter probably cannot be located. A possible solution to this problem is to adopt the new 4-cronered-trust model that was recently published in the X.509 (2016) standard [23].

REFERENCES

[1] S. E. Schechter, R. Dhamija, A. Ozment, et I. Fischer, « Emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies », in *In Proceedings of the 2007 IEEE Symposium on Security and Privacy*, 2007.

[2] A. S. Wazan, R. Laborde, D. W. Chadwick, F. Barrere, and A. Benzekri, "Which Web Browsers Process SSL Certificates in a Standardized Way?," in *Emerging Challenges for Security, Privacy and Trust*, 2009.

[3] ITU-T Recommendation X.509 | ISO/IEC 9594-8: "Information Technology—Open Sys- tems Interconnection-The Directory: Public-Key and Attribute Certificate Frameworks".

[4] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, 2008.

[5] P. Saint-Andre and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)," RFC 6125, 2011.

[6] https://adambard.com/blog/the-new-ssl-basics/

[7] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," RFC 4492, 2006.

[8] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," RFC 6960, 2013.

[9] A. Deacon and R. Hurst, "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments," RFC 5019, 2007.

[10] https://www.imperialviolet.org/2014/04/19/revchecking.html

[11] https://www.imperialviolet.org/2012/02/05/crlsets.html

[12] http://heartbleed.com

[13] http://www.zdnet.com/article/chrome-does-certificate-revocation-better/

[14] https://news.netcraft.com/archives/2014/04/18/chrome-users-oblivious-to-heartbleed-revocation-tsunami.html

[15] https://blog.mozilla.org/security/2015/11/23/improving-revocation-ocsp-must-staple-and-short-lived-certificates/

[16] Y. Pettersen, "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension," RFC 6961, 2013.

[17] D. E. 3rd, "Transport Layer Security (TLS) Extensions: Extension Definitions," RFC 6066, 2011.

[18] P. Hallam-Baker, "X.509v3 Transport Layer Security (TLS) Feature Extension," RFC 7633, 2015.

[19] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness," in *Proceedings of the 18th Conference on USENIX Security Symposium*, Berkeley, CA, USA, 2009.

[20] A. Jøsang, I. G. Pedersen, and D. Povey, "PKI Seeks a Trusting Relationship," in *Information Security and Privacy*, 2000, pp. 191–205.

[21] https://downloads.globalsign.com/acton/attachment/2674/f-06d2/1/-/-/-/-/globalsign-incident-report-13-oct-2016.pdf

[22] https://static.googleusercontent.com/media/pki.goog/en//GTS-RP.pdf

[23] A.S. Wazan, R. Laborde, D. W. Chadwick, et al., "Trust Management for Public Key Infrastructures: Implementing the X.509 Trust Broker," Security and Communication Networks, vol. 2017, 2017.