Digital Signature Trust Vulnerability:

A new attack on digital signatures

By Francesco Buccafurri

This paper describes a recently discovered vulnerability of the trust mechanism used to generate and verify digitally signed documents.

This paper describes a recently discovered vulnerability of the trust mechanism used to generate and verify digitally signed documents.[1] A digital signature is obtained from the original document by encrypting the hash digest of the document using a private key of an asymmetric cryptographic algorithm. Any change mechanism which does not modify the bits composing the document does not affect the validity of its digital signature, which in fact guarantees the integrity of the document itself (intended as integrity of its bits).

How digital signing works

The process of digitally signing a given file, say `letter.pdf`, consists of the following steps:

- The user opens the file `letter.pdf` using the signing software
- The software first computes a 160-bit *digest* of `letter.pdf`, sends it to the smartcard which encrypts it using the RSA algorithm with the user's private key, thus obtaining the digital signature
- The software receives the signature from the smartcard and creates a new file `letter.pdf.p7m` that is a *digital envelope* containing the original file `letter.pdf`, the digital signature and further information (including a public key X.509 *certificate*) that are needed by the verification steps

In order to verify the digitally signed file `letter.pdf`, the user must:

- Open the digital envelope `letter.pdf.p7m`, using the verification software
- The verification software first checks if the certificate is still valid
- Then the software computes the 160-bit digest of `letter.pdf` and compares it with the result of the decryption of the digital signature obtained by RSA and the user's public key (this is contained into the certificate)
- The success of the verification process depends on the result of the above comparison

Finally, observe that the verification software also allows the user to display the document in order to check the information signed by the subscriber. In our example, the verification software shows the document `letter.pdf`.

Digital signature ambiguity

Let us start with the description of an attack which is based on the possibility of a document having an ambiguous presentation without invalidating its digital signature: the displayed content may be different than the signed original, but signature remains validated. This ambiguity of content is well-known in the literature[2] [3] as a result of the inclusion of

1 F. Buccafurri, G. Caminiti, and G. Lax, "Signing the Document Content is not Enough: A new Attack on Digital Signature," in *Proc. of the IEEE International Conference on the Applications of Digital Information and Web Technologies* (ICADIWT 2008), August 4-6, 2008, Ostrava, Czech Republic, pp. 520-525, IEEE Press. The reader may find some more information about this result as well as a demonstration of the attack at http://www.unirc.it/firma.

2 A. Alsaid and C.J. Mitchell, "Dynamic content attacks on digital signatures," *Information Management & Computer Security*, 2005, 13(4), pp.328-336, Emerald Group Publishing Limited.

3 A. Spalka, A.B. Cremers, and H. Langweg, "Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs," in *Proc. of The Sixteenth Annual Working Conference on Information Security (IFIP/Sec'01)*, June 11-13, 2001, Paris, France, pp.403-420, Kluwer.

dynamic content (e.g., macros or JavaScript) in documents, whereas the new attack does not exploit this feature.

A document containing embedded instructions is not static: the visualization of its content might vary as the variables, which these instructions exploit, change. For example, suppose that a contract includes a dollar amount that is displayed as a result of a macro-instruction conditioned by the system date in such a way that after a given date the amount is changed. Obviously, in this case the bits of the digital contract do not vary, so the digital signature validation does not detect any modification event though the displayed content is different after the date than before the date, affecting the document in terms of the information it represents. Kain et al.[4] fully describe the problem and give some examples using MS Word, MS Excel, and PDF files.

A different source of ambiguity was discussed by Josang et al.[5] who show how font substitution can be used to display the same digital document with different meanings on different computers. The heuristics suggested to overcome this type of vulnerability consist of manually disabling dynamic content, working only with static file formats, and checking the document with a parser to automatically remove dynamic contents.

The *What You See Is What You Sign* concept[6] is designed to solve the ambiguity problem arising from signing digital documents with dynamic content. This approach works by creating a graphical representation – bitmap, tiff – of the digital document and then digitally signing it, a static image with no dynamic content.

File formant attack

This new attack does not rely on the ambiguity mechanisms described above but exploits a vulnerability of the digital signing process in which changing the signed file's name does not invalidate the signature. This represents an insidious vulnerability since it applies to file formats, e.g., images, that are traditionally considered safe.

Consider the following example: Prof. Buccafurri wants to delegate his assistant, Mario Rossi, to sign sales contracts on behalf of himself with amounts below $1,000. Prof. Buccafurri commissions Mr. Rossi to produce the electronic document to sign. In order to avoid possible insertion of macro-instructions or executable code into the document, Prof. Buccafurri asks his assistant to provide the document as a scan of a printed document (an image file, like bitmap or tiff). Note that before uncovering the new modality of attack here described, Prof. Buccafurri's precaution would have been considered prudent and safe. See Figure 1.


Delegation

I delegate Mr. Mario Rossi to sign sales contracts on behalf of myself with amount below $1,000 (one thousand) until 1 May 2008.

Reggio Calabria, 1 January 2008

Signature
Francesco Buccafurri

Figure 1 – The original document

At this point, Mr. Rossi, who is actually a thief, modifies the bits of this scanned image in order to carry out the attack. Next, the he saves the image as `delegation.bmp` and sends it to Prof. Buccafurri to be signed. Prof. Buccafurri signs the document by means of his smart card that produces the document's digital signature. This is sent from the smart card to the signature software in order to generate the cryptographic message in PKCS#7 format whose filename is `delegation.bmp.p7m` (a signed document is contained in a digital envelopment including the digital signature and all the material needed for its verification). Since it has been correctly generated and no alteration has been done, the signature verification of this document will clearly succeed.

Now, Mr. Rossi completes the attack simply by changing the filename of the cryptographic message from `delegation.bmp.p7m` to `delegation.htm.p7m`. Since the signature verification is done only on the basis of the bit stream included in the PKCS message, which does not contain the message filename, its change does not affect the result of the signature verification – the execution of the verification software on the renamed file succeeds. However, upon clicking the *display document* button of the signature verification tool (which allows us to see the signed document), surprisingly the document appears dramatically different from the signed one (see Figure 2). It contains a text giving Mr. Rossi the authority to sign sales contracts in the amount below $100,000, instead of the $1,000 approved by Prof. Buccafurri.


Delegation

I delegate Mr. Mario Rossi to sign sales contracts on behalf of myself with amount below $100,000 (one hundred thousand) until 1 May 2008.

Reggio Calabria, 1 January 2008

Signature
Francesco Buccafurri

Figure 2 – The tampered document

The vulnerability

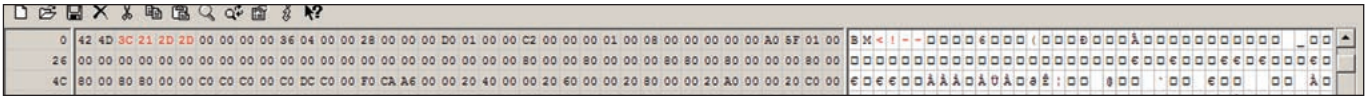
Which kind of vulnerability is exploited by the attacker? From a general point of view traditional paper documents are directly observable since they can be interpreted by humans using their senses (viewing and touching the document), mediated only by their capability of understanding the information contained in the document. Digital documents, however, are

4 K. Kain, S.W. Smith, and R. Asokan, "Digital signatures and electronic documents: a cautionary tale," iIn *Proc. of The Sixth Joint Working Conference on Communications and Multimedia Security*, September 26-27, 2002, Portoroz, Slovenia, pp.293-308, Kluwer.

5 A. Jøsang, D. Povey, and A. Ho, "What You See is Not Always What You Sign." in *Proc. of the Australian Unix User Group Symposium (AUUG2002)*, Melbourne, 4-6 September, 2002.

6 K. Scheibelhofer, "What You See Is What You Sign - Trustworthy Display of XML Documents for Signing and Verification," in *Proc. of Proceedings of the International Conference on Communications and Multimedia Security Issues*, May 21-22, 2001, Darmstadt, Germany, Kluwer.

Figure 3 – Hexdump of delegation.bmp after changing the header



not similarly observable in the sense that the bits composing them become meaningful to humans only when correctly interpreted by an application and presented through an analog physical medium such as a computer screen. This is a direct consequence of the immateriality of electronic documents on the one hand, and of its machine-level essence on the other hand, not allowing us to directly observe and understand their content by merely viewing a stream of bits. The consequence is that a digital signature may be both immaterial and support-independent, like the document content itself, so that it has to be necessarily linked to the bits composing the document. Indeed, complex instruments (interpreting applications) are needed for not only viewing the document but signing and verifying it as well. Thus, we cannot exclude the possibility of the bits composing the document of producing ambiguous tangible contents if interpreted by different applications. This is in fact the nature of the attack.

How it works

Let's delve into more technical detail about how the attack works in case of bitmap files. The tampered file includes at the same time two different contents – `.bmp` and `.htm` – with different encodes, each enabled by the application suitable for the respective format. The technique is quite simple. Consider the bitmap file `delegation.bmp` representing the original document. Understanding the bitmap format, the attacker modifies some bytes of the file by inserting an opening HTML comment `<!--` that will be skipped by any HTML interpreter (see Figure 3). Then, he creates a suitable HTML file (see Figure 4) including a given text *T*, which is the whole text of the delegation but with the new dollar amount. The HTML file has to begin with a closing HTML comment `-->`. At this point, the attacker appends this HTML file (after obscuring the text by using escape sequences) to the tampered picture file (see Figure 5). The resulting file is correctly opened by a bitmap viewer that shows the original text. Whenever the filename extension is changed from `.bmp` to `.htm`, the file will be opened by the browser showing the new text, which includes the modified amount (see Figures 6 and 7 next page). This tampering scheme can be applied to many other formats like `.pdf`, `.tif`, `.ps`, and `.rtf` – all implemented and tested.

```
-->
<HTML>
 <BODY>
  <STYLE>#l1 {background-color:#FFFFFF; POSITION:absolute;
        VISIBILITY:visible; TOP:0px; LEFT:0px; Z-INDEX:1}
  </STYLE>
  <DIV ID="l1"><P><center><B>Delegation</B></center>
    </P> I delegate Mr. Mario Rossi to sign sales contracts
     on behalf of myself <BR> with amount below $100,000
     (one hundred thousand) until 1 May 2008.
     <P>Reggio Calabria, 1 January 2008</P>
     <p ALIGN="RIGHT">Signature     
         <br>Francesco Buccafurri</P>
  </DIV>
 </BODY>
</HTML>
```

Figure 4 – HTML file with the malicious content

What happens when Prof. Buccafurri signs the file? The first step of the signature generation process is the computation on the document to sign of a cryptographic hash function (like SHA-1 or RIPEMD-160). The resulting digest (typically 160 bits) of the document and the properties of hash function guarantee that it can substitute the original document in the signature generation process since the probability of having two distinct documents producing the same digest is very low. Consequently, the problem of finding a document colliding on a digest of another distinct document is unfeasible, so that an attacker cannot corrupt a signed document without the signature detecting it. The digest is computed on the PC by the signature software (typically supplied by the certification authority) and sent to the smart card embedding the private key of an asymmetric cryptographic cipher, typically RSA. The smart card is then enabled by the user (by inserting a secret PIN) to encrypt the digest by RSA with the private key, thus producing the digital signature. It is finally sent from the smart card to the signature software running on the PC in order to produce the cryptographic message (typically in PKCS#7 format), `delegation.bmp.p7m` in our example.

PKCS#7 is a standard defined as RFC 2315 describing a general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes. PKCS#7 and X.509, which describes the format of the public key certificate granted by a trusted certification authority and aimed to indicate the real-life identity of the signer, guarantee the interoperability of software for verifying signed documents. The verification of document *D* is done by (1) re-computing the digest *I* of the document *D* with the hash function used in the signature generation process (this information is included in the PKCS#7 message), (2) computing *J* as the result of the decryption of the signature *F* done by means of the same algorithm as the generation step (as indicated in the PKCS#7 message) with the public key of the subscriber (included in

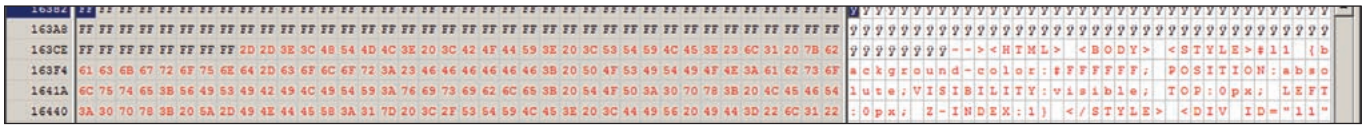


Figure 5 – Hexdump of delegation.bmp after appending the obscured HTML file

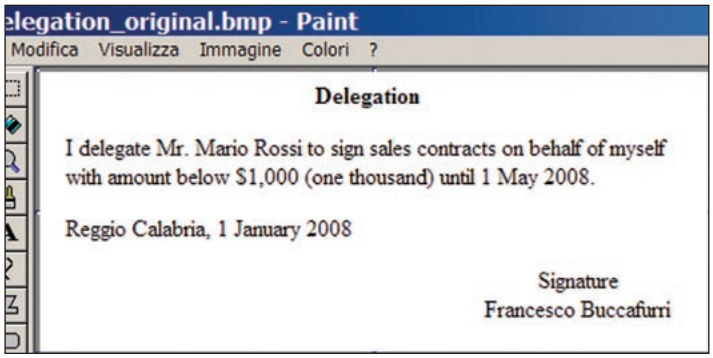


Figure 6 – The result of opening delegation.bmp

Figure 7 – The result of opening delegation.htm

the X.509 field of the certificate, which is another component of the PKCS#7 message), and (3) checking that the decrypted digest *J* coincides with the computed digest *I*.

Clearly, the complete verification has to check both validity, trustworthiness and non-revocation of the certificate, but we do not focus on this step since it is not involved in the attack here presented.

The output of the signature generation process is thus the cryptographic message named `delegation.bmp.p7m` (with double file extension). Now, if the attacker simply changes the filename of the cryptographic message from `delegation.bmp.p7m` to `delegation.htm.p7m`, the signature verification clearly succeeds, since no modification of the cryptographic message has been done. The interesting thing is that on most operating systems – Microsoft Windows, Linux with KDE, BSD with KDE, Mac OSX, etc. – the file extension determines the application used for opening the document. Whenever the document embedded into `delegation.htm.p7m` is displayed by the verification software, it will use the browser (because the name of the document is now `delegation.htm`), showing the document containing the modified dollar amount. Now the attacker owns a delegation from Prof. Buccafurri dramatically more powerful that the original one, allowing him to sign contracts to the higher amount, possibly with a company which is fraudulently related to him. As it can be easily recognized, by considering the verification steps described above, the fake contract appears as a legal document with a valid digital signature.

Prevention

What about solutions? A possible simple solution could be the inclusion of the document file name (or better the MIME-type) into the PKCS#7 cryptographic message by encoding it into the optional set of PKCS#9-compliant attributes that are signed together with the document. This way, the modification of the document extension is detected during the verification phase and the attack is disarmed. However, a technically simple solution does not correspond in general to a practically simple solution. Indeed, in this case, what about technical rules about the usage of PKCS#7 in the signature process? What about signature generation and verification software currently provided by certification authorities? Maybe a less simple, more heuristic solution could be more feasible, like a parser-based approach allowing us to detect patterns identifying the attack.

Conclusion

The Italian National Agency for Digital Administration (CNIPA)[7] issues the technical regulations concerning both the devices and the document formats to be used with digital signatures[8]; thus, we expect that, at least in the Italian case, some decision will be adopted in the near future. Indeed, the Italian CNIPA has considered the results presented in this paper very significant, claiming the intention of addressing the problem herein discussed in the preparation of the revised technical rules also by submitting these results to the Forum of European Supervisory Authorities for Electronic Signatures,[9] of which CNIPA is member. Interestingly, there was a significant echo about this result in Italy, both in Web discussions and in *Panorama*, a national weekly magazine. Therein, the journalist Luca Dello Iacovo relates the attack to some paintings of the famous painter Salvator Dalì, like *The Image Disappears* (1938) in which a somewhat mysterious picture of a bearded man (probably Dalì himself) and, on closer sight, a scene with a woman appear: Dalì's moustache is her arm, his eye is her head and his beard is her skirt. This comparison is highly appreciated since it succeeds in the hard task of giving charm and mystery to something that is devoid. It would be nice to recover something of this effect by calling the attack "The Dalì Attack."

About the Author

Francesco Buccafurri (PhD) is a full professor of computer science at the University of Reggio Calabria, Italy. His research interests include knowledge-representation, model-checking, information-security, data-compression, data-streams, and P2P-systems. He has published several papers in top-level international journals and conference proceedings. Professor Buccafurri may be reached at bucca@unirc.it.

7 CNIPA – http://www.cnipa.gov.it.

8 DPCM 13/01/2004 - Italian technical rules, http://www.cnipa.gov.it/site/_files/DPCM%20040113_v2.pdf.

9 FESA – http://www.fesa.eu.