Malware Attacks on Electronic Signatures Revisited

Hanno Langweg

Norwegian Information Security Laboratory – NISlab
Department of Computer Science and Media Technology, Gjøvik University College
P.O. Box 191, 2802 Gjøvik, Norway
hanno.langweg@hig.no

Abstract: We give an overview of the level of resistance current programs for the creation of electronic signatures show against attacks by malicious software ("malware"). Compared to tests completed five years ago, protection has gained strength. Yet, there is room for improvement.

1 Introduction

Electronic signatures still await their anticipated breakthrough. They pledge to alleviate authentication problems especially faced in B2C commerce and e-government. Automated mass signatures are not applicable in these cases, so people will use their personal computers to sign data. Malicious software on these computers (e.g., spyware) is a pressing problem and is not expected to abate soon. Five years ago, we investigated the level of protection software for electronic signatures offered against malicious software attacks. In this paper we explore what has changed in the time passed since.

We first provide an overview of previous and related work in this area. Section three lays out the attack scenario, defines metrics for grading resistance, and details our six attack categories. It is followed by the test results of several current programs, and a comparison. Finally we suggest methods to counter the attacks described in this paper on a structural basis.

2 Previous and related work

Resistance of signature software products against malicious software attacks has been investigated earlier by [SCL02]. The general challenges of signature creation on untrusted platforms are discussed in [Po93] and [Fo98]. Semantic problems of what has been signed are analysed in [JPH02] and [Po00]. Security requirements for signature creation applications [Ce01] related to malicious processes are stated simply: “All untrusted system and application processes, peripherals and communications channels that are not necessary for the operation of the SCA shall be prevented from interfering with the signing processes.” This has not changed in the proposed revision. [Ce04]
The only structured attempt so far to classify operating environments according to severity of expected malicious software threats is the result of a workshop lead by the German Bundesnetzagentur. [Bu05a] Operating systems of mobile devices are found to be unsuitable for the creation of electronic signatures by [MR04]. Creation of electronic signatures as developed in the SISI project [Bu05b] is supposed to be significantly more secure than previous approaches. Unfortunately, results of that project have not yet been published as of this writing.

3 Types of conducted attacks

With respect to [Bu05a], we consider attacks applicable to the environment called “Geschützter Einsatzbereich” (protected working environment). This is a reasonable choice as it is envisioned as the standard case and almost all products certified for use with qualified electronic signatures have been evaluated in view of this level. Relevant in our case are attacks by malicious software (via communication networks or by removable media) executed on a personal computer. Under this assumption and in accordance with [Bu05a], we expect constructive-technical measures of the signature software to mitigate attacks. These are generally based on the mechanisms available from the operating system, in our case Microsoft Windows XP Professional SP2.

We investigate attacks for which an expert attacker invests less than a day and uses no specialised equipment to find architectural vulnerabilities in the software as it was purchased. Exploitation should be possible automatically by laymen with appropriate tools. In terms of the Common Evaluation Methodology [Is04] we hence operate with an attack potential at level “Low”. We require neither physical presence nor possession of the signature card, and we assume some programming skills. Special knowledge of the signature software is not needed, apart from what is publicly available or what is obtained from having the product at hand. Privileges for executing the resulting malicious programs are determined by the respective user and are usually those assigned to the subject associated with that user. No “administrator” or “super user” privileges are required for exploitation.

We use three scales to grade resistance of applications against malware attacks. a) Primary attack method: misuse by user – accidental user cooperation – remote control of application – execution of arbitrary code. b) Primary attack result: attack not possible and detected – not possible – possible and detected – possible and undetected. c) Automation: attack requires customising by human – attack can be fully automated.

Data to be signed is prepared with an application program and then typically transferred to a signature creation application (SCA). The SCA offers the signatory to present the data to be signed and after confirmation transmits data to the signature creation device (SCD), usually a smart card. The SCD receives a personal identification number (PIN) – either via the SCA or via the card terminal – to authenticate the signatory and verify their presence. Data is then transmitted from the SCA to the SCD that computes the signature. The signature is sent back to the SCA and can then be stored together with the data used as input to the signing process.
All vulnerabilities examined by us are architectural weaknesses, i.e. weaknesses in the
design of a product. It is not required that there exist errors in the implementation of the
software, e.g. a buffer overflow vulnerability. Results can hence be used to compare
products and to be applied to improving software architectures in the future.

3.1 Modifying the data to be signed: An attacker might alter data after it has been
worked with in the application software and before it is used by the SCA. This can
happen at different opportunities: inside the application software as active document
content, during transfer between application software and SCA, while data is in storage,
when data is imported to the SCA. Often, a SCA uses a viewer component that presents
the data to be signed just before it is sent to the SCD. The signatory can check whether
the signature is applied to the correct data and proceed with or abort the operation. This
viewer is especially helpful when malicious code is present in the system. In a system
without malware a viewer would only prevent the user from accidentally signing the
wrong data. We check if data can be manipulated before it reaches the SCA and, if the
SCA employs a viewer, if presentation in the viewer can be modified by easy means.

3.2 Capturing the PIN to access the SCD: In an earlier survey, capturing the PIN to
access the SCD (i.e., the smart card), was a common vulnerability. The SCD usually
cannot determine if the signatory enters the PIN directly or if it is the SCA supplying the
PIN. Class 2 card terminals have a pin pad and can upon instruction transmit a PIN
entered at the pad directly to the card. In that case, the PIN is never made known to
processes on the personal computer and cannot be intercepted. We check if a SCA
supports PIN entry at the card terminal. If it does not or if it supports an alternative input
method on the personal computer, we check if it is possible for other processes to learn
about the PIN reliably and in short time.

3.3 Signing different data than intended: This threat is similar to introducing modified
data into the signing process as described in 3.1. The SCD cannot distinguish which
process sends the data to be signed. We check if communication with the card terminal
can be interrupted before secure PIN entry is activated and how the SCA reacts to this.

3.4 Interfering with the communication between SCA and SCD: We check if a driver
software manipulation is detected by the SCA and, if not, if secure messaging is used to
detect data flow manipulations by cryptographic means.

3.5 Executing arbitrary code in the SCA’s address space: An attacker gaining access to
the SCA’s address space might be able to introduce arbitrary code running with the
privileges of the SCA. Execution of the SCA would then be completely untrustworthy as
there are no means to reliably detect and defeat malicious code in the same process. We
check if the standard installation sets reasonable access rights, if the SCA detects
unreasonable access rights (i.e. full control for everyone), if modules of the SCA can be
modified or replaced without detection, if card terminal driver modules can be modified
or replaced without detection.

3.6 Modifying the signed data to be verified: We check if a viewer is employed for
signature verification and if the presentation in the viewer can be modified by easy
means to hide a manipulation.
4 Surveyed products

Selection of products is based on the following criteria. Applications have to offer electronic signatures on desktop personal computers in connection with a smart card. We prefer software and smart cards that are evaluated and certified for use as signature creation applications or secure signature creation devices, respectively. Included are products from Germany and Austria as of autumn 2005. We tried to re-test some products that we had in our earlier test. Most of them have been discontinued or have been replaced by different versions. We use a range of signature cards and card terminals; details are provided in the following sections. It is not intended to give a comprehensive test of all available and suitable products. Our selection is limited by availability of products, cards, card terminals, and budget. Yet, we feel the collection covers most of the most popular applications (if this is possible in a market still small and without distinct major players).

4.1 Deutsche Telekom T-Telesec Signet 1.6.0.4

Signet is technically based on the SignCubes software and sold through the business outlets of Telekom’s T-Punkt shops. This year, the first clerk we asked knew that the product existed. He was unsure whether it was really sold by the local shop or if it had to be ordered centrally. Twenty minutes of internal hot-line conversation and a search of the inventory yielded the desired box. We already had a card for electronic signatures issued by Telesec’s trust centre. Nevertheless, the software was bundled with another card. Registration failed because of a misspelled server address; we had to return later. This time registration worked. When prompted to choose a revocation password we discovered that the AutoComplete feature of the local browser was enabled. Hence, our password was stored in the cache of the computer at the shop. Disabling this feature was not possible owing to restrictions in the profile of the local user. Three passwords of previous signature card customers were stored. The clerk mentioned that they probably sold three or four copies of Signet per year at that location. Installation was without problems. Access rights were set according to permissions of the parent folders. We used the context menu extension to sign files out of the Windows Explorer.

Data can of course be modified after working with it in an application and before selecting “sign”. This is countered with Signet’s secure viewer module – or so it seemed. Using Windows messages the Signet software can be made to display a different application, application executable name, and data file name (cf. Figure 1). The dialog window is consistently brought to the foreground, should another application try to get the focus. However, it is possible to place a borderless inactive window at the top of the z-order, containing only a button that imitates the dialog’s “show data in secure viewer” button. If the user clicks on that button, Signet’s viewer can be executed by the malicious software and given a completely different file as a command line parameter. Hence, the user signs data different from that presented in the viewer. In addition, it is possible to draw on the viewer’s presentation surface.
The PIN is never entered at the personal computer during the signature process. However, when we activated the signature PIN for the first time, we were forced to do it at the computer, because the option of using the class 2 card reader was not available. When changing the PIN, secure PIN entry is possible, but not as default.

We put a modified WINSCARD.DLL (for PC/SC card reader communication) in Signet’s folder. While all of Signet’s modules are signed and hence could be verified upon loading and execution, this operating system file is not. Apparently the library file is not referenced by its full path (it usually is in %WINSYSDIR%). It is located first in the folder where the calling application resides. To be able to add the file, we had to be a member of the Power Users group, a condition not met by all attackers, but far from requiring administrator status. Signet loaded and executed our DLL giving access to its address space and permitting arbitrary malicious actions. We did not check whether we were able to explicitly monitor or manipulate communication with the card as this is only a special case after gaining access to the running program.

### 4.2 IT Solution trustDesk standard 1.2.0

TrustDesk was ordered by email and phone and arrived a week later. No hassles. We opted for the expensive class 3 Siemens Sign@tor card terminal. Our shipment was a worn box with the card terminal and a CD-R. The included viewer trustview is recommended by the Austrian trust centre a.trust. Installation was challenging. We installed the card reader first, followed by trustDesk. The installer/trustDesk detected the driver for the (unplugged) SCM SPR532 we had used before. We had to choose that reader to be able to proceed. Later we switched to the Siemens reader. During installation we got a scary warning by the operating system that a new certificate was going to be installed. As told in the message we called the issuer (IT Solution GmbH) to verify the certificate’s thumbprint. It took them several minutes to finally point us to a server with the posted thumbprint. Access rights were set according to parent folders’ permissions. We used the context menu extension to sign files out of Windows Explorer.
Data can of course be modified after working with it in an application and before selecting “sign”. This should be countered with trustDesk’s secure viewer module. Nevertheless, it is possible to draw on the viewer’s presentation surface. Since the user can freely enter the expert configuration mode and no special permissions are set for the corresponding registry keys in the user tree, it is possible for malicious programs to modify the card reader configuration. Hence a fake driver library file can be specified. Upon accessing the card reader trustDesk loads and executes our modified device driver giving access to the software’s address space. We did not check whether we were able to explicitly monitor or manipulate communication with the card as this is only a special case after gaining access to the running program.

4.3 D-Sign matrix/digiSeal 3.0.1

DigiSeal was sent from D-Trust as a replacement for their D-Sign matrix software which was not on stocks when we ordered. They said that it was an identical product. Installation was without problems. Access rights were set according to permissions of the parent folders. We used the virtual printer to sign documents out of applications.

Data can be modified after working with it in most applications and before selecting “sign” (e.g. by active content or macros). This is countered with digiSeal’s secure viewer module – or so it seemed. It is possible to modify the viewer’s presentation surface without detection. Using Windows messages the digiSeal software can be made to display a different verification result, signature creation time etc. The PIN is never entered at the personal computer during the signature process. However, secure PIN entry is not the default option and has to be configured by an administrator first. The user (or a malicious program acting on the user’s behalf) can change the reader configuration and specify a new card terminal driver. Thereby it is possible to load arbitrary malicious code in the address space of digiSeal. We did not check whether we were able to explicitly monitor or manipulate communication with the card as this is only a special case after gaining access to the running program.

4.4 Ventasoft venta-sign 2.0.0.968

We obtained the software from the manufacturer’s homepage. Installation was without problems. Access rights were set according to permissions of the parent folders. We used the context menu extension to sign files out of the Windows Explorer.

Data can of course be modified after working with it in an application and before selecting “sign”. This is countered with venta-sign’s external viewer option – wait, there is no viewer option. The product does not employ its own viewer and does not link to an external one. It is possible to draw on the application’s presentation surface, thereby showing the user a different file name and file information. Same modification is applicable to signature verification results and to results of the integrity checker software. Instead of the common Windows controls, venta-sign uses Qt widget controls. This makes it harder for malicious software to manipulate presentation as it lacks a convenient interface.
Secure PIN entry is supported. If unsecure PIN entry is chosen by the user, a warning is displayed. Keylogging remains possible, obtaining the PIN via messages sent to the application does not succeed. We did not succeed in persuading the SCA to use our modified WINSCARD.DLL. Apparently it is loaded from the Windows System folder where access permissions prevent modifying access for ordinary users. Placement in the installation folder did not help. We therefore did not check whether communication with the card terminal could be interfered with, because this would have required successfully executing our modified library file. The manual indicates that a failed signature creation (e.g., by forging card communication) will lead to a security warning issued to the local user.

4.5 2B Secure FILE 1.0

We obtained the software in the shareware version “2B Secure Bundle Version 1.0”. The online shop of the manufacturer looked like it had not been updated for some time with its many broken links. The shareware version bears some restrictions, but is fully functional as regards creating qualified electronic signatures with SignTrust cards. It is probably based on Deutsche Post’s discontinued SignTrust mail application. User interface and hints that it is based on an evaluated product suggest this ancestry. Installation was without problems. Access rights were set according to permissions of the parent folders. We used the context menu extension to sign files out of the Windows Explorer.

Data can of course be modified after working with it in an application and before selecting “sign”. This is countered with SecureFILE’s external viewer option – or so it seemed. Using Windows messages the SecureFILE software can be made to display a different viewer application, data file name, and data folder. In fact, only date/time, selected certificate, and security level are refreshed regularly; all other information in the window can be changed maliciously. The desktop surface seems to be disabled while signing. This hinders the user to start other applications by clicking on their icons (running applications and the Start menu are not affected). It is possible to place a button that imitates the dialog’s “execute viewer” button, similar to our experience with Signet. If the user clicks on that button, an arbitrary viewer can be executed by the malicious software. Hence, the user signs data different from that presented in the viewer. Our class 2 card reader supported secure PIN entry, SecureFILE did not. It however prevented capturing the PIN by simple means, e.g., Windows messages. Using a keylogger would have yielded the PIN entered at the keyboard. We did not succeed in persuading the SCA to use our modified WINSCARD.DLL. Apparently it is loaded from the Windows System folder where access permissions prevent modifying access for ordinary users. Placement in the installation folder did not help. SecureFILE probably uses the full path to the library file. Another explanation is that the software’s structure (a composition of DLLs initially loaded by explorer.exe) leads to a search path where system folders are looked at first. We did not check whether communication with the card terminal could be interfered with, because this would have required successfully executing our modified library file.
4.6 Utimaco SafeGuard Sign & Crypt for Office 3.4.1

Sign & Crypt passed our elementary tests well last time, so we wanted to include it this time. Getting a copy of SafeGuard Sign & Crypt proved to be difficult. The fourth distributor we had contact with delivered it after almost four weeks as a trial version. Installation required an updated installer and a restart. Access rights were set according to permissions of the parent folders. We used the context menu extension to sign files out of the Windows Explorer. The secure viewer sometimes threw a runtime exception, and tests were impeded by not having a fully compatible smart card. None of the smart cards of the 'large' trust centres was supported.

Presentation in the secure viewer component was easily alterable. This surprised us, considering that Sign & Crypt's viewer had been the strongest in our previous tests. A card reader with a PIN pad was supported, secure PIN entry, however, was not configurable and the PIN could thus be recorded on the computer. We were unable to replace the SafeGuard smart card provider, and hence could not interfere with card communication or execute malicious code in the address space of the signature application.

4.7 Comparison

Test results for all surveyed products are shown in Table 1. The first column lists product and version, the second indicates if it is a product certified for use with qualified electronic signatures. The following columns refer to the attacks performed.

<table>
<thead>
<tr>
<th>Product</th>
<th>Cert. prod.</th>
<th>Modify DTBS</th>
<th>Capture PIN</th>
<th>Forge DTBS</th>
<th>Comm. SCA–SCD</th>
<th>Execute code</th>
<th>Modify DTBV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signet 1.6.0.4</td>
<td>yes</td>
<td>yes</td>
<td>no*</td>
<td>n/t</td>
<td>n/t</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>trustDesk standard 1.2.0</td>
<td>yes*</td>
<td>yes</td>
<td>no*</td>
<td>n/t</td>
<td>n/t</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>digiSeal 3.0.1</td>
<td>no</td>
<td>yes</td>
<td>no*</td>
<td>n/t</td>
<td>n/t</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>venta-sign 2.0.0.968</td>
<td>yes</td>
<td>yes</td>
<td>no*</td>
<td>no*</td>
<td>no</td>
<td>no</td>
<td>no*</td>
</tr>
<tr>
<td>2B Secure FILE 1.0</td>
<td>no*</td>
<td>yes</td>
<td>no*</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no*</td>
</tr>
<tr>
<td>Sign &amp; Crypt 3.4.1</td>
<td>no</td>
<td>yes</td>
<td>no*</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

DTBS – data to be signed; DTBV – data to be verified; SCA – signature creation application; SCD – signature creation device; n/t – not tested; * – see text
It comes at no surprise that all programs are vulnerable to modification of DTBS before starting the signing process. However, even where a secure viewer is provided, the presentation in the viewer can easily be forged by malicious software. Capturing the PIN is not possible if secure PIN entry is used; this method is supported by all tested products. The first three products allow execution of externally provided (malicious) code. Here, arbitrary data could be signed. Three products defeat this attack method, probably by organisational means (access permissions set for operating system files). Using our metrics from chapter 3 we see that Signet, trustDesk, and digiSeal allow for the most rewarding attack method, while venta-sign, Secure FILE, and Sign & Crypt are limited to accidental user-cooperation and remote control of the application. All attacks go undetected; venta-sign might detect an aborted signature creation and give a warning. All attacks can be fully automated without customisation during exploitation.

If we compare manuals and certifications for the tested products we find that they assume a protected operating environment completely free of untrustworthy programs. In most cases, constructive-technical protection measures comprise an integrity check of SCA modules on demand and secure PIN entry at the card terminal. Our attacks do not require administrator permissions and do not change the operating system. The only change introduced to the operating environment is a program using the standard operating system application programming interface.

5 Protection against the presented attacks

5.1 Targeted countermeasures

Modifying the data to be signed: There is a perceived need for a viewer component. This viewer is necessary primarily because of modifications made to data before it is submitted to the signing process. Avoiding manipulation at this stage could involve a tight coupling of the data processing application and the signature application. Another approach is to make the data fixed before it becomes obvious to an attacker that the data is relevant for signing. A method to conceal this would be the use of a write once read multiple medium, e.g., as discussed in [SCL02].

Capturing the PIN to access the SCD: A typical countermeasure (and employed by the surveyed applications) involves a card terminal with a pin pad. The PIN entered by the user is sent directly to the card and cannot be captured by processes running on the personal computer. However, secure PIN entry can also be emulated without a class 2 reader if a Trusted path mechanism is available to the signature application. Secure PIN entry should be the default option when a supporting terminal is detected.
Signing different data than intended: A Trusted path between the application and the local human user can alleviate this threat. Authenticity and integrity of program output is needed for a viewer component, and authenticity and integrity of user input is needed for confirmation of actions. The Trusted path mechanism is known from operating systems security. Cost-effective methods to retrofit applications are detailed in [La04]. PIN entry at the personal computer provides a common secret shared among the SCA and the SCD. This is lost when PIN entry is done directly at the card terminal. The SCA no longer can authenticate itself to the SCD. A simple solution to this problem would require changes to the cards (and card issuers are unlikely to like it). The basic idea is to use secure PIN entry twice to authorise a signature creation. First PIN entry opens a signing session. Data to be signed is sent to the SCD. The SCA can check if the correct data has been sent and if so, prompt the user to securely enter the PIN again. That closes the signing session. If communication with the card is aborted just before the first PIN entry and fraudulent data is sent to the card, the SCA can issue a warning. The user would have to enter the PIN twice, and a single accidental PIN input would not lead to a signature.

Interfering with the communication between SCA and SCD: Standard solution is to use secure messaging for the communication with the signature creation device.

Executing arbitrary code in the SCA’s address space: None of the products surveyed makes explicit use of Windows XP’s access control mechanisms. It is put into the responsibility of the user/administrator to configure access control permissions correctly. Integrity of installed modules can be verified algorithmically, e.g., by a tool delivered on the installation medium of the software. This tool could also check the access control configuration. Modules loaded and executed by the SCA should be checked automatically before use in the SCA’s address space. This is straightforward with modules distributed by the SCA’s manufacturer. Integrity and location of system modules, e.g., card terminal device drivers and APIs, might require a regularly updated list. However, software automatically checks for updates, so it should be feasible to check for a “white list” of known good third party files as well. Use of the SigAll-API [Si05] could ameliorate this situation. A SCA could also be split into an untrusted part and a separate trustworthy application accessible only via few defined interfaces.

Modifying the signed data to be verified: Similar considerations apply to presentation of signature verification results as to presentation of data to be signed. Additionally, the semantics of the signed data need to be determined unambiguously. This is outside the scope of this article.

5.2 Trusted Computing platform

Ideas how to use Trusted Computing technology in connection with signature creation devices exist. [SCL01] In addition, the usual mechanisms like verification of module authenticity and integrity, separation of processes, and a Trusted path between the local human user and the signature creation application would be beneficial. Still, all this hinges on support by the operating system. In our case (the Microsoft Windows line of operating systems) first plans by the manufacturer looked promising. Alas, newer developments tend to reduce expectations.
5.3 SISI (Sichere SignaturInfrastruktur)

After weaknesses found in commercial software for the creation of electronic signatures some years ago, BSI and TC TrustCenter AG in 2003 started a project “SISI – Sichere SignaturInfrastruktur”. The aim was to establish “trustworthy conditions for the creation of signatures in standard PC environments”. [Tc03] Results claimed to be delivered include that “[c]omponent providers will gain access to important information and suggestions while users will be able to use trustworthy products that are reliably protected against manipulation, for example viruses or trojan horses, by state-of-the-art technologies”. Unfortunately, the only published result about this project so far is the notion of its completion that appears in [Bu05b]. That source states that – by help of SISI – users are enabled to run electronic signature applications on standard operating systems (e.g. Microsoft Windows XP or SuSE Linux 9.0) in a way that manipulations by malicious code are detected or even prevented.

5.4 SigAll-API

The approach of the SigAll-API [Si05] is to abstract away differences among signature cards and card terminals for application developers. Applications and their users are provided with a uniform interface regardless of the specific cards and terminals involved. We encounter the following challenges as regards malicious software attacks. Communication with the card and PIN entry is transferred from signature applications to the SigAll-Factory, Card Providers, and Reader Providers. Assuming that there are more signature applications than card and reader providers, these security-relevant mechanisms are implemented less often and can be evaluated more easily. Device drivers, reader providers, and card providers could be tightly coupled. However, responsibilities are undefined with respect to checking integrity and authenticity of card and reader modules. SigAll-API components could become attractive targets for writers of malicious code, since they ensure compatibility with all signature applications.

6 Conclusions

More than eight years after the legal introduction of electronic signatures it is still not easy to obtain a certificate, signature creation device, and software. Compared with tests performed five years ago, products have improved. Support for secure PIN entry at the card terminal prevents capturing of the user’s PIN by malicious applications. Signing of executable modules and verification of module signatures is employed more often. Access to system modules (i.e. card terminal driver libraries) depends less on organisational measures (i.e. the operating system’s default search method).
All applications lack a Trusted path or a similar mechanism to ensure that presentation of data in the viewer or in dialogs during the signing process cannot be manipulated by malicious software. All applications depend on organisational measures as regards access permissions to their modules. No application checks if access permissions are set reasonably after installation. Some applications allow the user to choose card terminal driver libraries and hence allow arbitrary code to be executed in their address space. No revolutionary new mechanisms are needed to prevent or detect attacks. Some security measures discussed in section 5 are already incorporated in some products.

The test results give an indication of the resistance of an application against malicious software attacks. However, our tests are not meant to replace thorough examination of a product in an evaluation scheme, e.g., the Common Criteria. It must neither be construed as a proof of absence of vulnerabilities nor as an endorsement of a specific product.

References