TRANSPARENT ACCESS TO ENCRYPTED DATA USING OPERATING SYSTEM NETWORK STACK EXTENSIONS

Ero Rademer (Fraunhofer-IGD, Darmstadt, Germany)
and Stephen D. Wolthusen (Fraunhofer-IGD, Darmstadt, Germany)

Abstract The CIPRESS system provides security enhancements for general purpose operating systems by adding kernel level functionality for cryptographic and steganographic operations and keeping both users and application programs unmolested as far as possible. This paper describes the transparent network filtering and encryption mechanisms used in the Microsoft Windows NT implementation that allow integrated access and use control over confidential or otherwise restricted data at client systems.

1. INTRODUCTION

Relying on users to handle data security by themselves is a risky proposition. Even if users are completely trustworthy, the probability that necessary operations are omitted (e.g. asserting access control lists on local systems, using encryption mechanisms) due to inconvenience or lack of time is always present; this problem exists even with people that by most standards should know better (Snider and Seikaly, 2000). In addition, virtually all analyses of attacker profiles in recent years indicate that a significant part of the IT security breaches is caused by current or former employees.

Any organization wishing to secure confidential or classified information – or to ensure that intellectual property rights are honored – must therefore use stronger mechanisms enforcing security constraints even against the will of the users if necessary. At the same time such security measures should be unobtrusive.

Another problem that is typically ignored is the handling of use control. If one stipulates that users cannot be fully trusted with the material obtained, then transmitting files for local use or even for viewing on a browser-type application creates an opportunity to use the data in an unauthorized way. Even if

the initial access was controlled, most systems do not deal with use control at all; once a user has access to the data, further re-use (e.g. by copying and pasting to other documents) is not addressed. Traditionally, this is dealt with only in applications and therefore subject to a number of attacks and functional limitations.

However, even if use control mechanisms are implemented without the possibility for circumvention, some problems remain unsolved, chiefly that at some point the document is transformed into an analog representation (e.g. on screen, as a printout, as an analog audio output,...). This representation can then be used to circumvent the most elaborate encryption and access control mechanisms by simply removing a printout from the premises. While prevention is not feasible (even if printing were disallowed, the issue of screen shots remains), there is a possible deterrence mechanism, namely the use of digital watermarking to identify both the owner of a document and the user who accessed the document. In conjunction with the abovementioned automated cryptographic mechanism for access and use control, these watermarks can also be embedded at the kernel level without the possibility of user intervention.

The CIPRESS[1] system is an attempt at addressing this conundrum by adding mandatory access control, encryption, and auditing mechanisms to existing operating environments which provide adequate support for both applications and security extensions. This paper describes the part of the CIPRESS system that provides transparent network layer access control, auditing and encryption on the Microsoft Windows NT platform. The system was implemented under Microsoft Windows NT 4.0 starting in the spring of 1998, a first working prototype including the foundation of the network filtering and encryption mechanisms described here was released in 1998; the result of ongoing research during 1999 and early 2000 is described here.

1.1. BASIC CONCEPTS OF CIPRESS

CIPRESS adds security mechanisms at the kernel level and thus is able to enforce a security policy for all applications and users while being largely invisible to users and applications; the kernel/user mode separation is also used for protecting itself; similar mechanism have been used by others (Jones, 1993; Reynolds and Heller, 1991) to add non-standard functionality to existing operating systems.

The enforcement of the security policy is ensured by keeping protected data mandatorily encrypted[2] at all times and decrypting it only on the fly and after verification of access rights without the possibility of intervention by the user. Key material is not stored locally on client systems but rather forwarded from a central trusted site (the Key Center) to a trusted environment[3] on the

client system wishing to perform the encryption (store) or decryption (load) operation[4]. Since verifying each file system access via a centralized database would be highly impractical, distinctions are made between general data (encrypted using a common key specific to a machine or user called the MasterKey) and documents intended for exchanging between individual systems or users. Only for the latter so-called registered documents[5] the central Key Center access control and key granting mechanism is used. After a user has logged into the system and authorized himself to the Key Center, all operations regarding key handling are performed transparently and without further user interaction. Documents are identified by a cryptographic hash[6] value; the Key Center allocates a symmetrical key to each $\langle \mathrm{document}, \mathrm{user} \rangle$ tuple $\langle D_i, U_j \rangle$. Since documents are transmitted only ever in encrypted form, each document access requires a permission verification and key request – this results in an effective means for achieving use control, not just access control since the enforcement mechanism stays intact beyond the initial access; furthermore, a request by a user $U_{j'}$ to store a registered document D_i obtained from user U_j will automatically result in the document being encrypted with the specific key $\langle D_i, U_{j'} \rangle$ and an audit log of the access being generated. Due to the chaining of the encryption keys (this mechanism is called ReEncryption), a tree of a document's propagagation can be generated at the Key Center. In any case the en- and decryption on the fly is realized both at the file system and at the network level in an operating system extension, so that the appearance of the system to both users and applications is that no change compared to an insecure system is detectable; only someone e.g. attempting to access a file system without the CIPRESS system running will see only encrypted data.

This mechanism goes beyond merely using labels for enforcing access control in at least one important area; in a networked environment the possibility exists that an individual system can get compromised since there is no such thing as perfect security. In case of a local security breach (e.g. by compromising the host operating system) an attacker will find only encrypted data on the attacked system. Compromised systems can be blacklisted so that neither operational clients nor, more importantly, the Key Center will communicate with insecure systems. Furthermore, changes in access control lists on registered documents take effect immediately regardless of the storage location, while the system contains central repositories also used for registration purposes called Content Server systems, it is completely irrelevant for the operation of the encryption mechanism whether the document is retrieved on-line from a Content Server or e.g. read from an archive on a DVD-ROM or simply the local hard disk.

CIPRESS is not a multilevel security system in the traditional sense; it merely enforces the access and use control mechanisms on registered documents. The remaining component of the security concept that facilitates the move beyond

rigid compartmented levels is that of tainting. Files created or merely touched by a user with write access are automatically encrypted as MasterKey documents even if they are plaintext files located on a remote file system. Only files which match a cryptographic hash of a registered document may be exchanged, and for these the Key Center enforces the security by granting or denying keys. A user may therefore create new documents or copy and paste from a registered document to which she has access; it can only be forwarded to other users by registering the newly created file with the Key Center. As noted in section 4, this behavior dovetails with the requirements of a typical Microsoft Windows NT working environment where a separation into compartments is rather difficult. However, if an operating environment and applications support such behavior, enforcing compartmentalization of tainted files becomes feasible using the mechanisms present in the CIPRESS system, and one can easily implement a multilevel security system with the appropriate security policy in place.

As noted before, no encryption system can protect against the possibility that a document is legitimately obtained and then converted to an analog representation only to be removed. As in the case of encrypting files and network traffic, it is irrelevant whether a legitimate copy falls into the hands of an adversary due to an oversight by a legitimate user or if the illegitimate removal of the analog copy is done deliberately. CIPRESS attempts to address this issue by embedding a digital watermarking (Busch et al., 1999) mechanism in the operating system alongside the encryption mechanisms, that is, a mechanism for embedding secret (or public) information in the carrier signal of a multimedia document in such a way that it is difficult to remove or tamper with even though it is not perceptible. This mechanism ensures that any registered document for which a watermarking mechanism exists, a watermark identifying the user that retrieved the document (i.e. the user for which the Key Center has granted the decryption key and logged the document access) is embedded into the document. This occurs regardless of the type of data access (i.e. from a file system or over a network connection) and takes place before the application and hence the user has access to the document. Any printout or screen shot therefore contains the identity of the user; in addition to that, the digital watermark developed by our group is capable of supporting hierarchical digital watermarks, the document therefore also contains two additional watermarks which are already embedded at the time of document registration. One of these two server watermarks is a secret watermark known only to the administrator of the Content Server at which the document was registered (typically this role belongs to an organizational security administrator) and the operator of the Key Center. The other watermark is a public watermark that can be read by anyone with the appropriate tool and allows the identification of the original (digital) document from the analog representation by extracting an identity code for the source Content Server and a sufficiently large (48 bits) fragment of the SHA-1

cryptographic hash. This allows one to identify the digital source document even if only a fragment[7] is available. This can be of value in and of itself withhout considering the security aspects.

The overall system architecture is beyond the scope of this discussion; for a by now somewhat dated general overview see (Busch et al., 2000).

2. THE MICROSOFT WINDOWS NT NETWORK STACK ARCHITECTURE

Due to the requirements for a basic security model described in section 1.1, the following discussion is limited to NT-based operating system family; this family includes Microsoft Windows NT 4.0 and Microsoft Windows 2000; earlier versions of Microsoft Windows NT did not have the WinSock $2.x$ API introduced with Microsoft Windows NT 4.0 although it is possible to retrofit it on Microsoft Windows 95. While there have been a number of changes in Microsoft Windows 2000, the discussion here applies to both releases.

Microsoft Windows NT provides several networking APIs, namely

- WinSock
- Named Pipes
- Mailslots
- Remote Procedure Call
- NetBIOS
- Telephony

Other services such as DCOM may be layered on top of these interfaces; while some of these interfaces have their own security and encryption mechanisms (such as RPC) others rely on the connection being assumed as secure and simply enforce access controls (e.g. named pipes and mail slots which are implemented as file systems and can use the access control mechanisms for file systems, see (Solomon, 1998; Solomon and Russinovich, 2000).

To obtain a reasonably secure network configuration, transport bindings must be restricted. While it would be possible to modify transport mechanisms such as IPX/SPX, AppleTalk, VINES IP, and NetBEUI for supporting access control and encryption, it is hardly worth the effort since in most cases IP can be substituted; besides, such a mechanism would lead to a proprietary protocol stack. Apart from such considerations it should also be noted that each transport mechanism supported brings with it the potential for fatal security flaws in its implementation, so in this as in many other cases, less is more.

Furthermore, some interfaces such as Telephony are notoriously insecure since they bypass the usual protection mechanisms and must be disabled in a secure environment.

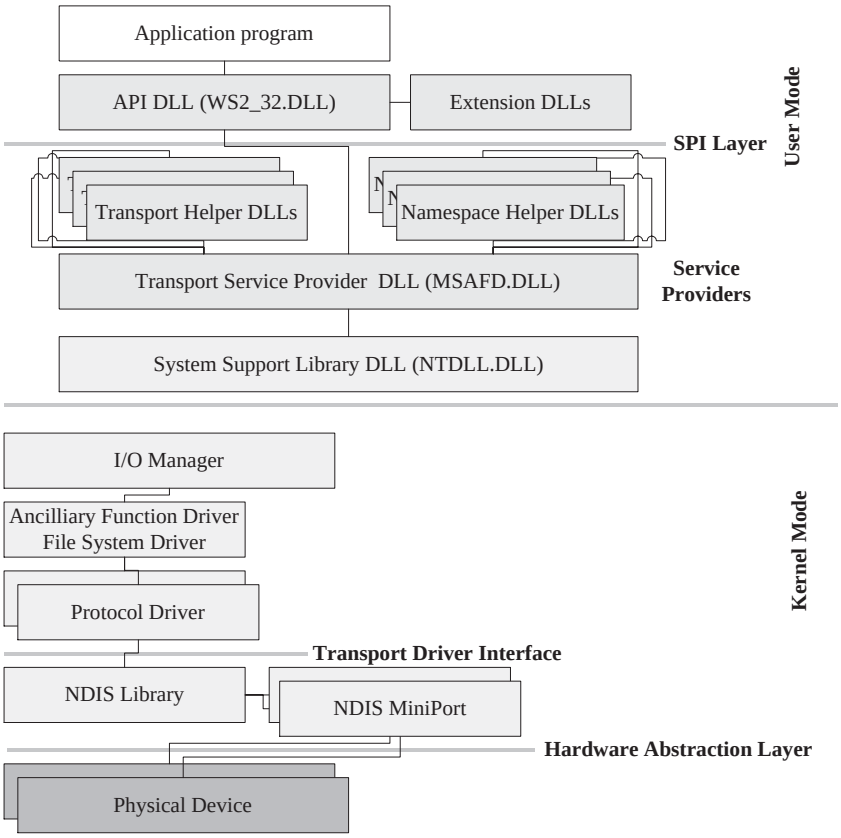

Figure 1 Interacting Components in WinSock Calls

Assuming that one concentrates on using IP as the sole transport mechanism, one can then implement a security mechanism by interposing such a mechanism at the service provider level (see figure 1).

The network architecture of Microsoft Windows NT consists of a number of layers. At the lowest level is the physical device, access to the device is regulated by the hardware abstraction layer (HAL). Device drivers are realized as NDIS (Network Driver Interface Specification) modules consisting of the generic NDIS library and the device-specific NDIS miniport drivers; the library fully encapsulates the miniport drivers. Accessing the NDIS library is the TDI (Transport Driver Interface) mechanism. This itself consists of transports (or protocol drivers), supporting the various transport mechanisms such as NetBEUI and TCP/IP, and TDI clients which provide services for sockets and NetBIOS calls. None of these modules can be called directly from appli-

cations since they are protected kernel mode interfaces. Upper-level APIs such as NetBIOS and Windows Sockets are implemented at the user level.

The Windows Sockets API (or WinSock) is modeled after the original BSD sockets (McKusick et al., 1996) and has undergone significant revisions under various platforms before arriving in its current form (Andersen, 1997a; Andersen, 1997b). It is available for both the NT-based and DOS-based operating systems from Microsoft Corporation.

It consists itself of several modules. From an application's perspective the sockets API consists of the exposed API DLL (dynamic-link library); this DLL communicates with the SPI (Service Provider Interface) layer. This layer is controlled by the transport service provider DLL which in turn calls on a number of transport helper DLLs and namespace helper DLLs to perfom its mission. On the other hand, the transport service provider DLL forwards the thus generated calls to the System Support Library DLL that represents the interface to the abovementioned kernel components. Since the Microsoft Windows NT design is predicated on a file system model and represents sockets as file handles, a translation mechanism is required. This service is performed by an Ancilliary Function Driver (AFD).

Of particular interest in this is the ability to stack several of the transport helper DLLs so as to provide additional services at each level (there is no layering mechanism for namespace helper DLLs). WinSock here distinguishes between "base protocols" and "layered protocols". The former are protocols capable of performing actual communication with a remote endpoint, the latter must rely on base protocols for actual communication and only provide added value. At least in theory it is possible to implement several stacked layers of such layered protocols, permitting the implementation of a variety of services.

3. IMPLEMENTATION

At the time of the initial implementation in 1998 the documentation of the layered service provider (LSP) mechanism was limited to (Andersen, 1997b), since then (Butterklee et al., 1999) has appeared, and both Microsoft Windows 98 and Microsoft Windows 2000 are now using a LSP to implement quality of service (QoS). It therefore appears that others have recognized the validity of the approach taken for implementing the security mechanisms in CIPRESS for Microsoft Windows NT.

Logically the provider can be separated into two parts. One is responsible for providing a secure channel while the other is tasked with handling the registered documents. These two functions are largely independent and can be layered themselves with the secure channel layer being at the bottom of the stack.

However, our experience has shown that inserting more than one provider into the stack is rather problematic. The majority of these problems occur in conjunction with Overlapped I/O, that is also involved in other issues as described below.

A Layered Service Provider must announce itself to the WinSock by registering a new Provider Catalog Entry. This catalog entry can be considered the "new layer". For the WinSock subsystem to actually know when the new layer has to be called and where in the service provider hierarchy it has to be placed additional Service Provider Chain Catalog Entries have to be made. In our case this is done for the TCP and the UDP protocol; although technically only TCP monitoring is required, it is necessary to do so for UDP as well, otherwise an error condition in the operating system is triggered. To ensure that the WinSock Service Provider cannot be registered twice, each of the Provider Catalog Entries is assigned a unique catalog number (GUID). This mechanism as well as the deregistration is part of the provider and can only be called by a thread with administrative privileges.

Although the WinSock Service Provider will work for most applications and the Microsoft Windows NT system services itself when performed on an Microsoft Windows NT 4.0 system with Service Pack 3 installed, the official documentation states that a Layered Service Provider must make use of a system call which was not provided up to Service Pack 4. This affected all socket operations which involved Overlapped I/O such as Microsoft ODBC and might be one reason why this interface was used so sparingly until very recently. For the service provider to work properly, the WinSock elements from Service Pack 4 or above had to be present on a Microsoft Windows NT 4.0 system. Later releases and Microsoft Windows 2000 do not suffer from this omission.

As described in (Andersen, 1997b), the service provider DLL must be registered with WinSock. Loading the DLL is performed by WinSock itself, not by user applications, all uses of WinSock after the installation will be forced to operate through this layer.

3.1. THE ENCRYPTION LAYER

For providing the secure channel we selected the SSL (TLS) (Frier et al., 1996; Dierks and Allen, 1999) protocol. While there are certain drawbacks to this such as the limitation to TCP-based connections and the need for elaborate session caching to obtain acceptable performance, there are some advantages to this mechanism. One of these is that the protocol is well understood with the core elements remaining stable for a long time. These core elements include some cipher suites and the handling of mutual authentication based on public-key certificates.

The protocol should not be confused with the often haphazard implementation found in a number of products, especially WWW browsers and servers that tend to be rather lenient in both the cipher suite negotiation and, more importantly, in honoring the authentication mechanism process. If implemented correctly – i.e. with security instead of convenience as the objective – SSL can ensure strong ciphers and authentication. For the purposes of establishing secure channels (e.g. from a client to the Key Center), CIPRESS therefore requires that strong mutual authentication has succeeded based on certificates known to the provider (typically this would only be an organization's own CA, accepting arbitrary third-party certificates is not desirable[8]), and that an acceptable cipher suite has been negotiated. Otherwise establishment of connections is denied.

An important benefit of a SSL-based implementation is that it does not interfere significantly with network management mechanisms since it is operating at the application layer in the OSI reference model. This should allow easier integration of CIPRESS systems into existing network structures without necessitating new mechanisms and tools, as will be required once IPSEC mechanisms are deployed more widely.

Another possibility opened by the use of the SSL protocol is that one can use COTS accelerator subsystems (provided they can handle the required cipher suites[9]) instead of the software-based mechanism on server systems. Since the assumption here is that the server systems are in a trusted environment in any case, this generally should not represent a significant degradation of security.

3.2. NETWORK ACCESS CONTROL

During the installation process the WinSock Service Provider is provided with an access list describing for which hosts a secure channel must be established. This mechanism can also be used to communicate rules for packet filtering based on IP addresses and hosts[10] in cases where a restriction to communicating only with CIPRESS hosts is too restrictive[11]. This mechanism does not replace an external firewall (which is able to counter a number of attacks that cannot be dealt with in the location covered here), but it ensures that network connections created by applications are limited to those systems covered by the local security policy without the possbility for user manipulation.

3.3. HANDLING REGISTERED DOCUMENTS

The second, theoretically independent function of the WinSock Service Provider is to detect the transmission of registered documents and to provide encryption and decryption on the fly so applications and users will not notice. For this purpose, registered documents are prefixed with plaintext header in-

formation. The WinSock Service Provider forwards packets recieved from a socket unchanged to the calling application unless a “magic string” is detected in the data stream. In case of a detection, the WinSock Service Provider must read the remaining parts of an as yet hypothetical registered document without affecting the handling of false positives. It should be noted that false negatives have no security impact; in such a case a legitimate client would merely not receive the plaintext.

If the detected magic string is indeed the start of a registered document, the WinSock Service Provider must read the the rest of the header and the document body itself for further processing; otherwise it is the service provider’s responsibility to abort any read-ahead (in case that the received data indicates that there is no registered document in the data stream) as soon as possible and to give back the data gathered so far.

The criteria to detect a “this is not a proper header” condition are (in order of reception):

1 The magic string was not completed correctly
2 The data format for hypothetical fields do not match (e.g. binary vs. ASCII data required by the format)
3 The data length field has an invalid value (zero).
4 The CRC value (computed over the entire header) does not match.

In all cases, a (completed or not completed) header received in more than a particular number of chunks is rejected; also, there is a maximum timeframe during which the provider waits for any missing parts, either header or document data (both conditions would indicate an interactive session with someone typing the header data). Since a header may start anywhere within a data stream, a sliding window mechanism must be applied to search for the start of such a header.

Once detection was successful, the application does not receive any more data, instead the WinSock Service Provider gathers the necessary data off the network stream, allocates a protected memory mapping, and calls a system service that is also part of the CIPRESS system (and also running with administrative privileges, separatd from any user process) for further processing.

This processing begins with verifying the encrypted document’s integrity and requesting the ReEncryption key material from the Key Center; if this is not granted, the encrypted data is forwarded to the application and the provider continues searching for registered documents.

In case permission and key material was granted, the document is decrypted[12], and a user fingerprint digital watermark (i.e. an identification code for the user under whose identity the document was retrieved) is inserted if a watermarking mechanism is available and registered for a given data type inside the service. Only then is the flow of control returned to the WinSock Service

Provider which can now use the decrypted and (if possible) marked data from the memory mapping to feed into the application. To the application, this process appears largely transparent; the transmission behavior is changed in that a delay in transmission is followed by a burst of receiving activity. Overall transmission time is increased only by the latency introduced by the decryption and watermarking.

Since digital watermarking introduces noise into the carrier signal of the marked data, even systems as the one used here reach a limit regarding the number of markings that can be embedded in the same carrier signal. If such a threshold is reached, the service will automatically retreive another copy of the document from the Content Server that is home to the digital document and start a new sequence of markings with this fresh copy.

Both this automatic re-fetching and the abovementioned retrieval of the key material cause an interesting recursive condition in the internal state of the WinSock Service Provider and must be handled with extreme care. A similar situation occurs in case of a cache miss or implicit retrieval operation in the service described in section 3.4

3.4. DIRECTORY SERVICES

While mutual authentication according to the TLS standard requires that certificates are exchanged during each (re-)authentication, and issues such as expired certificates can be handled during the authentication stage, there is still a need for a directory service for revoked certificates.

This is handled by means of another system service, the Directory Service. Due to performance reasons it is not desirable to perform an online directory lookup every time a certificate is verified. Therefore the Directory Service contains a local cache which can store recently used certificate revocation lists and perform verification services for the WinSock Service Provider using a fast local interprocess communication mechanism (Local Procedure Calls). At certain intervals, new CRLs can be retrieved using a directory mechanism (LDAPv2 is used for this purpose; since only passive retrieval of signed data is required, no elaborate authentication mechanism is necessary). As discussed before, only a single CA should be used for performance reasons; this CA public key must be stored locally in the client in a secure storage area and updated if and when the CA public key is renewed.

3.5. PROBLEMS ENCOUNTERED

As noted before, the documentation of the interfaces used is not exactly adequate, and at least earlier operating system versions contained incomplete implementations of WinSock despite claims to the contrary. In addition to this, a number of applications caused problems by program errors which are appar-

ently ignored or even "supported" by the operating system-supplied provider such as attempting to read from a closed socket (file handle) and expecting to read valid data. Since one of the application exhibiting this problem is considered a major player, the erroneous behavior had to be supported in our implementation as well. Other problems surfaced in the previously mentioned Overlapped I/O mechanism, especially when using the Microsoft-specific extensions to the WinSock API.

4. LIMITATIONS

As noted in section 1.1, the tainting concept in CIPRESS implies that each file touched by a user with write permissions must be encrypted with a MasterKey. The way in which virtually all applications running under Microsoft Windows NT are implemented results in the need for allocating a dedicated workstation to each individual user; there is typically no clear separation of data belonging to an individual user and write-protected data owned by a separate (system) account for the application[13]. This is not a limitation of the CIPRESS system but rather the result of faulty application design. However, users require the functionality in any case and dedicating a system to a user is much less expensive than creating properly designed applications with the features of e.g. office productivity applications.

To ensure confinement of restricted data, the system must take a conservative stance. While it can control incoming data, outgoing communication always carries the risk of leakage or covert channels. The network security mechanism must therefore ensure that all communication is encrypted and restricted to other valid CIPRESS nodes. Merely encrypting the transmission between nodes is insufficient since unless the target node is also a fully operational CIPRESS node one cannot be ensured that the access and use control mechanisms are also enforced on the receiving end.

The network stack security extension implementation on Microsoft Windows NT also results in the facility being available only for the Win32 subsystem, the OS/2 and POSIX subsystems (Solomon, 1998) must be disabled since they do not access the WinSock mechanism. However, given the number of applications and non-existent support for the latter subsystems this should not be of grave concern.

5. OUTLOOK

We are currently implementing both Microsoft Windows 2000 and Sun Solaris releases in addition to the Microsoft Windows NT 4.0 prototype. There are a number of areas in which the security and usability of the CIPRESS system in general and the network filtering mechanims in particular can be enhanced, and to which future research should be directed, e.g. the integration of

additional information on the subjects participating in the process and to further restrict certain types of privileged communication to a set of applications which have been evaluated for their security and conformance to the organization's policy. Another potential enhancement to the system is the support for streaming data; currently only complete files are handled.

The authors would like to thank the Mitsubishi Corporation, Tokyo, Japan for its generous support and close cooperation in supporting this research.

Notes

1. CIPRESS (Cryptographic Intellectual Property Rights Enforcement SyStem) is an internal project code name of Mitsubishi Corporation and Fraunhofer-IGD

2. Currently Triple DES (National Institute of Standards and Technology (U. S.), 1994) with three keys is used for all encryption operations

3. CIPRESS supports hardware extensions for this purpose. However, a pure software version is also available but is limited in security by the possibility of tampering with the software environment by skilled aversaries.

4. The term "client" denotes the relationship to the CIPRESS servers – such systems can be servers themselves in other contexts

5. The term document denotes arbitrary data in this context

6. Currently SHA-1 (National Institute for Standards and Technology (U. S.), 1995) is used for this purpose

7. Experiments have shown that a 10% fragment of a printout is sufficient for the recovery of the full watermark

8. This is mainly due to performance consideration. Even with aggressive session caching, following chains of certificate places an undue delay on the establishment of connections

9. RSA/3DES/SHA-1

10. Microsoft Windows 2000 introduces an undocumented feature for packet filtering that might be an alternative if the packet filtering API is published or at least acknowledged to exist on all versions of this and future generations of the operating system family

11. This might e.g. be the case if certain infrastructure services such as DNS are permitted, but all other communication is restricted to a secure network

12. The current system again uses 3DES CBC

13. The encryption of files shared between users due to the application design would result in the first user touching such a file being encrypted with that user's key, rendering it inaccessible to another user

References

Andersen, D. B. (1997a). Windows Sockets 2 Application Provider Interface. Technical report, Intel Corp. Version 2.2.1.

Andersen, D. B. (1997b). Windows Sockets 2 Service Provider Interface. Technical report, Intel Corp. Version 2.2.1.

Busch, C., Funk, W., and Wolthusen, S. (1999). Digital watermarking: From concepts to real-time video applications. *IEEE Computer Graphics and Applications*, 19(1):25–35.

Busch, C., Graf, F., Wolthusen, S., and Zeidler, A. (2000). A system for intellectual property protection. In *Proceedings of the World Multiconference*

on Systemics, Cybernetics, and Informatics (SCI 2000) /Int'l Conf. on Information Systems Analysis and Synthesis (ISAS 2000), Orlando, FL, pages 225–230.

Butterklee, B., Hua, W., and Ohlund, J. (1999). Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider. *Microsoft System Journal*.

Dierks, T. and Allen, C. (1999). RFC 2246: The TLS Protocol Version 1.0.

Frier, A., Karlton, P., and Kocher, P. (1996). The Secure Socket Layer (SSL) 3.0 Protocol. Technical report, Netscape Communications Corp.

Jones, M. B. (1993). Interposition agents: Transparently interposing user code at the system interface. In Liskov, B., editor, *Proceedings of the 14th Symposium on Operating Systems Principles*, pages 80–93, New York, NY, USA. ACM Press.

McKusick, M. K., Bostic, K., Karels, M. J., and Quarterman, J. S. (1996). *The Design and Implementation of the 4.4 BSD UNIX Operating System*. Addison-Wesley Publishing Company.

National Institute for Standards and Technology (U. S.) (1995). Secure Hash Standard (SHA). Federal information processing standards publication 180-1, NIST, Gaithersburg, MD, USA.

National Institute of Standards and Technology (U. S.) (1994). Data Encryption Standard (DES). Federal information processing standards publication 46-2, NIST, Gaithersburg, MD, USA. Supersedes FIPS PUB 46-1-1988 January 22.

Reynolds, F. and Heller, J. (1991). Kernel support for network protocol servers. In USENIX, editor, *Proceedings of the USENIX Mach Symposium: November 20–22, 1991, Monterey, California, USA*, pages 149–162, Berkeley, CA, USA. USENIX.

Snider, L. B. and Seikaly, D. S. (2000). Report on Investigation: Improper Handling of Classified Information by John M. Deutch. Central Intelligence Agency Inspector General Report 1998-0028-IG. Unclassified, FOUO.

Solomon, D. (1998). *Inside Windows NT*. Microsoft Press, Bellevue, WA, USA, 2nd edition.

Solomon, D. and Russinovich, M. (2000). *Inside Windows 2000*. Microsoft Press, Bellevue, WA, USA, 3rd edition.