Covert Channel Analysis and Detection using Reverse Proxy Servers

WJ Buchanan [A] and D Llamas [B]
School of Computing, Napier University, EH10 5DT, Scotland, UK

Keywords : Reverse Proxy Server, Covert Channel, Data Hiding
Words : 3380

Abstract

Data hiding methods can be used by intruders to communicate over open data channels (Wolf 1989; McHugh 1995; deVivo, deVivo et al. 1999), and can be used to overcome firewalls, and most other forms of network intrusion detection systems. In fact, most detection systems can detect hidden data in the payload, but struggle to cope with data hidden in the IP and TCP packet headers, or in the session layer protocol. This paper contains proposes a novel architecture for data hiding, and presents methods which can be used to detect the hidden data and prevent the use of covert channels for its transmission.

1 Introduction

A covert channel is a communication channel that allows two cooperating processes to transfer information in a manner that violates the system's security policy (Berg 1998). Basically, it is a way of communication that it is not part of the original design of the system, but can be used to transfer information to a process or user that, a priori, would not be authorised to access to that information. Covert channels only exist in systems with multilevel security (Proctor and Neumann 1992), those that contain and manage information with different sensitivity levels , so it allows different users to access to the same information at the same time but from different points of view, depending on their needs to know and their access privileges. The covert channel concept was introduced in 1973 (Lampson 1973) and, since then, several researches have been done about this attack system that specially affects to systems where the most important security aspect is data privacy.

Generally, covert channels are classified based on various aspects (Gligor 1993):

- **Scenarios:** in general, when building covert channels scenarios, there is a differentiation between storage and timing covert channels (Lipner 1975). The former are channels where one process uses direct (or indirect) data writing whilst another process reads this data. Generally, uses a finite system resource that is shared between entities with different privileges. On the other hand, the covert timing channels use the modulation of certain resources, such as the CPU timing, in order to exchange information between processes.
- **Noise:** as any other communication channel, covert channels can be noisy or immune to noise. Ideally, a channel immune to noise is such where the probability of the receiver listen exactly what the sender has transmitted is 1: there are no interferences in the transmission. Obviously, in real life it is very difficult to obtain these perfect channels hence it is common to apply error correction codes despite these reduce the wide band of the channel.
- **Information flows:** in the same way than in the conventional lines of transmission different techniques are applied to increase the wide band, something similar can be done in the covert channels. Channels where several information flows are transmitted between sender and receiver are denominated aggregated channels, and depending on

how sent variables are initialled, read and reset, aggregations can be classified as serial, parallel etc. Channels with a unique information flow are denominated non aggregated.

The concern for the presence of covert channels is, as mentioned above, common in high security systems such as military ones. In fact, many of the studies done about attacks based on covert channels and its prevention have been, and are being done, by United States governmental and military bodies (National Security Agency, US Air Force, National Computer Security Centre …). However, in other environments it is also possible the existence of covert channels, especially in protocols like the TCP/IP protocol suite (Route 1996; Rowland 1996).

On the other hand, an ordinary proxy server is an intermediate server that sits between the client and the origin server. In order to get content from the origin server, the client sends a request to the proxy naming the origin server as the target and the proxy then requests the content from the origin server and returns it to the client. The client must be specially configured to use the forward proxy to access other sites (Apache 2001).

A typical usage of a forward proxy is to provide Internet access to internal clients that are otherwise restricted by a firewall. The forward proxy can also use caching to reduce network usage.

A reverse proxy, by contrast, appears to the client just like an ordinary web server. No special configuration on the client is necessary. The client makes ordinary requests for content in the name-space of the reverse proxy. The reverse proxy then decides where to send those requests, and returns the content as if it was itself the origin (Apache 2001).

A typical usage of a reverse proxy is to provide Internet users access to a server that is behind a firewall.

2 Reverse Proxy Server to hide data communications

2.1 The current security context

In the security context nowadays, the control over the information transport mechanisms it is more important than ever in order to guarantee its correct operation under normal circumstances and also when attacks take place. This control is also required in order to ensure that these transport mechanisms, basically protocols although this could be extended to some other elements, are not used to hide information.

The ordinary proxy servers are extremely useful for the purposes mentioned above as they concentrate the traffic associated to the user, both incoming and outgoing, allowing the creation of different kinds of restriction rules, authentication rules etc.

The use of proxy forward servers implies a specific configuration in the client side, which in some ways assumes that the user is aware that his connection with external networks will be subject to some rules and controls. In the security context, information as basic as this one, might be very interesting for those who intend to use covert channels.

The reverse proxy servers, by contrast, appear to the client just like an ordinary service (for example, a web server). No special configuration on the client is required. Depending on the kind of reverse proxy server, the returned content can be as if it was itself the origin. This paper proposes a novel architecture for data hiding and detection throughout a Data Hiding Intelligent Agent embedded on a reverse proxy server. This agent is in charge of performing both, hiding and detection activities, as well as the prevention management and the application of countermeasures to the use of protocols as a mechanism of transport of information.

In the current security context, where almost anything can be considered as information, it is highly recommended the use of solutions based on stegano-components, meaning hidden components that can work in a discrete mode.

Although it is not the focus of this paper, there is a novel technique called Dynamic Reverse Proxy (DRP) where a dynamic connection is established between the user and the reverse proxy server, without requiring any configuration in the client side. This link, totally independent and with its own properties and methods, is in charge of the control and analysis of the traffic, always operating in a hiding way and in a mode totally discrete.

2.2 Data Hiding Intelligent Agent (DHIA)

The proposed architecture for the management of hidden information as well as the mechanism for its monitoring and detection, prevention and countermeasures activation is the Data Hiding Intelligent Agent embedded in a Reverse Proxy Server.

The prototype designed manages requests at the HTTP level but, as mentioned later on, the number of protocols will be increased in the future so the service provided is more powerful and complete.

The user thinks that is connecting with a web server and navigates on it. All this is totally transparent for the navigation.

The scenario and its components would be as follows:

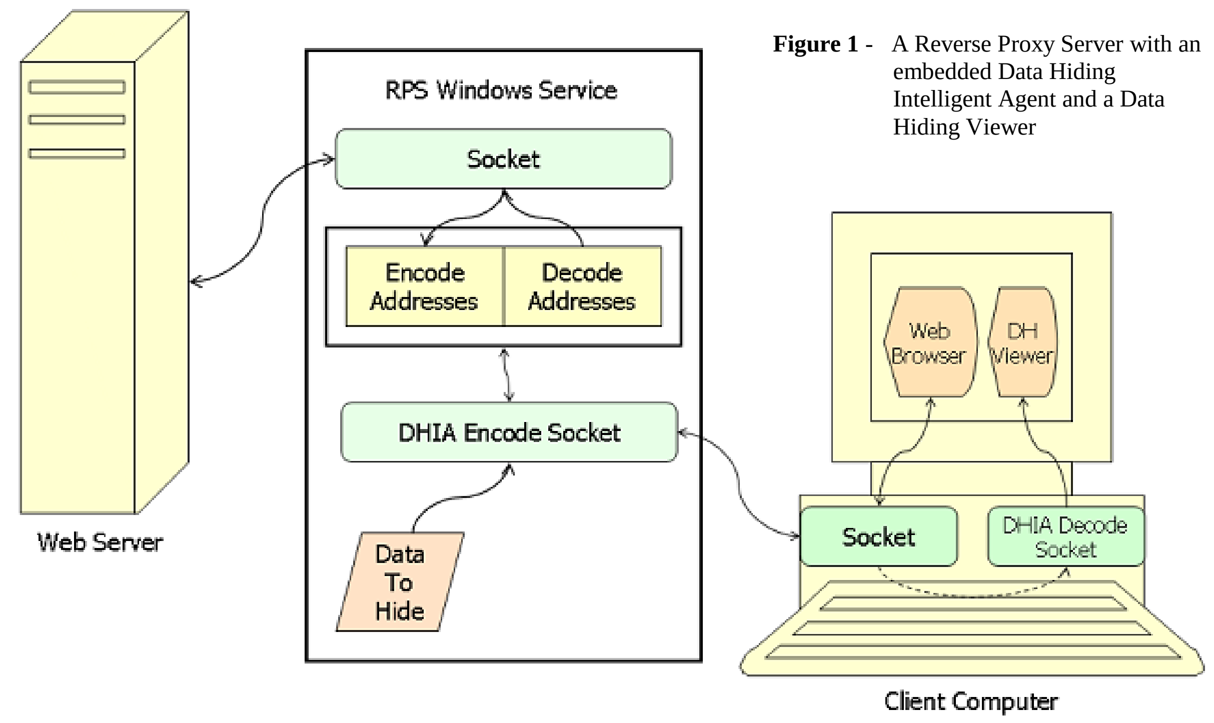

Figure 1 - A Reverse Proxy Server with an embedded Data Hiding Intelligent Agent and a Data Hiding Viewer

Three main pieces of software are identified in this scenario: A Reverse Proxy Server with an embedded Data Hiding Intelligent Agent and a Data Hiding Viewer. While the user is navigating around a web site through the Reverse Proxy Server, the Data Hiding Viewer on the client side will show the hidden data sent by the Data Hiding Intelligent Agent embedded in the Reverse Proxy Server.

The Data Hiding Intelligent Agent component is being tested in different positions within the scenario where it should operate, in order to find its more convenient situation, taking also into account the Dynamic Reverse Proxy technique.

About the specific information hiding technique used in this implementation, it is proposed in this paper the covert communication through the manipulation of the Identification field of the IP protocol header, version 4 (Rowland 1996). In this case, the implementation is done using the first byte as sequencer and the second byte to host the character in ASCII code.

The Identification field of the IP protocol header helps with the re-assembly of packet data by remote routers and host systems. Its purpose is to give a unique value to packets so if fragmentation occurs along a route, they can be accurately re-assembled.

2.3 Monitoring and detection

The monitoring is performed using the common method of placing the network card in promiscuous mode and sniffing the network traffic. Is well known that the own definition of covert channels, as well as the steganographical techniques, presumes a high level of difficulty relating to detection issues. The methods designed by the DHIA are those related with the detection of sequences in the fields of the headings of the different packets that circulate in the network. Other techniques such as the extra traffic payload sensoring, the analysis of encrypted content or the creation of packets ad-hoc has no effect.

2.4 Prevention and Countermeasures

The prevention and countermeasures for the communication based on covert channels will depend on the techniques from which protection is required. The use of the reverse proxy server as a middleware element implies a scenario based on a minimum of two connections, this means, the one done by the user from his computer to the reverse proxy server in a transparent way and the one done between the reverse proxy server and the web server which the user wants to navigate in. This system allows an easy control of the packets when they go from one segment to another, for example, doing an overwriting of the Identification field, which will automatically mean the elimination of the original content.

3 Implementation on Windows platforms

A novel aspect is the implementation of the covert channels technique in windows platforms. In all the windows operating systems, the TCP/IP protocol is propietary, and its source code is not accessible which means that the manipulation of the packets in any of the TCP/IP protocol suite is not possible from levels above the TCP/IP driver layer. This makes more complex the use of these techniques in windows platforms.

3.1 Windows NT/2000 Network subsystem architecture

Microsoft Windows NT/2000 network architecture is composed of software components that provide networking abilities to the Windows NT/2000 operating system. Network communication begins when an application program attempts to access resources on another computer. Data and requests move from layer to layer within a computer. Each layer is able to communicate with the layer immediately above it and the layer immediately below it. If the packet is not meant for use by the current layer, it passes the packet to an adjacent layer. Packets travel down the network protocol stack of the first computer. If the destination is on another networked computer, the packets of data are sent across the physical media (such as wiring, fiber optic cable or satellite). The data is passed upward through the lower layers of the second computer up to the same layer that started the exchange of data (Microsoft Corporation 2000).

On the other hand, all this is divided in two modes: Kernel and User

In Kernel mode, the processor can execute all instructions, including those designated “privileged”, and can access all of memory. The Kernel provides a set of services that the rest of system can use. It calls the Hardware Abstraction Layer to handle any necessary platform-specific operations.

In User mode, an application can access only the memory to which the operating system has granted it permission. A user-mode program can, of course, ask the operating system to please change the map, but it is the operating system, running in Kernel mode, that actually makes the change, if it decides the change is permissible.

See the following graphic for a global view of these modes and the affected drivers and components (Divine 2002):

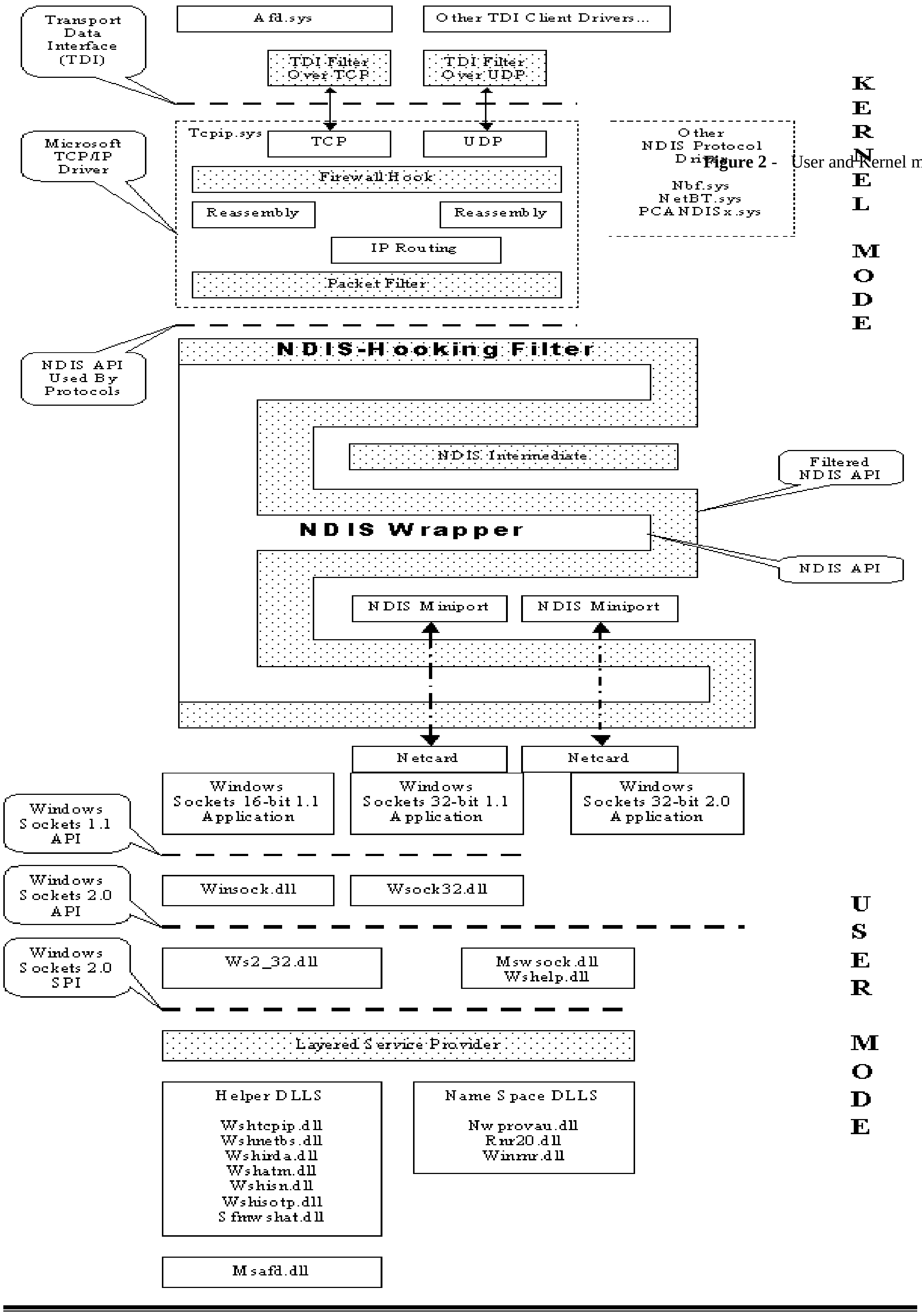

Figure 2 - User and Kernel modes

3.2 Network traffic filtering technologies for Windows

Possible ways for network traffic filtering (Divine 2002; V.Smirnov 2003)

User-mode traffic filtering.

- 1) ***Winsock Layered Service Provider (LSP)***. As method advantage it can be mentioned the possibility to determine process that called Windows Sockets. This can be used for such tasks like QOS (Quality Of Service), encryption of data streams and etc. However, don't forget that TCPIP can be called directly via TDI; therefore this method is in no use for Trojan/virus protection and etc. Additionally, this approach can't be used on router, because packets are routed on the TCPIP level (or even on MAC level, for details search in DDK for FFP).

- 2) ***Windows 2000 Packet Filtering Interface.*** Windows 2000 provides API using which user-mode application can install a set of "filter descriptors", which will be used by TCPIP for packet filtering (PASS/DROP). However, rules for filtering are rather limited (pass/drop based on IP address and port information), and this approach can be used only starting from Windows 2000 and higher, what is rather serious limitation.

- 3) ***Substitution of Winsock DLL.*** This approach mentioned only for security reasons. Not recommend.

- 4) ***Global hook of all "dangerous" functions (starting from Windows sockets, DeviceIoControl and etc)***. It can be done but it has possible impact on overall system stability and security.

Kernel-mode traffic filtering:

- 1) ***Kernel-mode sockets filter.*** This technology is applicable for Windows NT/2000. It is based on interception of all calls from msafd.dll (the lowest level user-mode Windows Sockets DLL) to the kernel-mode module afd.sys (the TDI-client, kernel-mode a part of Windows Sockets). This method is interesting, but its possibilities are no much wider, than LSP's. In addition I would like to add that AFD interface is extended from version to version of Windows NT family what limits it portability.

- 2) ***TDI-filter driver.*** This technology applied for both Windows 9x/ME and Windows NT/2000 though concrete implementations strongly differ. As for Windows NT/2000, in case of TCP/IP filtering it is necessary to intercept (using IoAttachDevice or patching dispatch table in driver object) all calls, directed to devices created by tcpip.sys driver (\Device\RawIp, \Device\Udp, \Device\Tcp, \Device\Ip, \Device\MULTICAST). This technology is well known and also it's used in a number of commercial products (for example, Outpost Firewall). However, as well as methods above, this approach can be used only for creation of personal firewall class products, and it can't protect you TCPIP stack from hackers attacks.

- 3) ***NDIS Intermediate Driver.*** Microsoft publicly introduced NDIS Intermediate drivers in NT 4.0 as a way for developers to write packet capture drivers. NDIS Intermediate drivers can see all network traffic taking place on a system because the drivers lie between protocol drivers and network drivers. Developers base software that provides fault-tolerant and load-balancing options for NICs on NDIS

Intermediate drivers. Actually, support of IM drivers was improved in Windows 2000/XP, but there is another problem, the driver must be digitally sign at Microsoft, elsewhere user will have a horrible nag-message.

- 4) ***Windows 2000 Filter-Hook Driver***. The Filter-Hook Driver was introduced by Microsoft in Windows 2000 DDK. In fact, it is not a new network driver class, it is only a way to extend IP Filter Driver included with Windows 2000 functionality.
- 5) ***NDIS Hooking Filter Driver.*** Briefly the technique based on interception of some subset of NDIS functions which in the further allow to trace registration of all protocols installed in the operating system and opening of network interfaces by them. Among advantages of the given approach it is necessary to mention ease of installation and transparent support of Dial-Up interfaces what require additional work in case of IM drivers. This technique is the one that has been used to implement the Data Hiding Intelligent Agent (DHIA).

3.3 NDIS Hooking Filter Driver

The Network Driver Interface Specification allows to hook into the network layer as ethernet packets are being passed to and from the Network Interface Card at the Windows Kernel mode. Through some API's, the interception of these packets can be finally done in the Windows User mode where most windows software runs. A C DLL was written to intercept and modify outgoing packets and to allow information to be reported to a controlling application. The controlling application was written with Microsoft Visual C#.NET 1.1.

Due to security in Windows and the .NET Framework raw sockets, as per the Berkely specification, are not fully supported. Therefore modifying packets at this level is not possible as the .NET Framework and the Windows operating system will correct any faults it perceives in the header or ignores the header completely and treats it as the payload. This is undesirable behaviour.

4 Conclusions and Future Work

The conclusion of this work is that the applications that can operate on a discrete mode, such as the reverse proxy servers, which are oriented to the monitoring, detection and prevention of the information hiding techniques such as covert channels, steganographical techniques an so on, will have a vital importance in the security context.

The use of Reverse Proxy servers as a transparent element and middleware component for high security environments would be required immediately, as the attack techniques in the information world develop very quickly.
It has been shown in this paper the importance of the Reverse Proxy server in the high security environments as well as the novel development of this kind of tools for windows platforms.

In future work, we are planning to perform more extensive experiments that will involve the inclusion of new covert channel techniques associated to the manipulation of the heading fields of any of the TCP/IP protocol suite (or other techniques that we are currently investigating such as the use of the TTL field) and an extension to the Reverse Proxy Server to manage requests at different levels.

5 References

Apache (2001). HTTP Server documentation. v. 1.3.

Berg, S. (1998). Glossary of Computer Security Terms, National Computer Security Center.

deVivo, M., G. O. deVivo, et al. (1999). "Internet Vulnerabilities Related to TCP/IP." SIGCOMM Computer Communication Review **29**.

Divine, T. F. (2002). Windows Network Data and Packet Filtering. I. Printing Communications Assoc.

Gligor, V. D. (1993). A Guide to understanding Covert Channel Analysis of Trusted Systems, National Computer Security Cente.

Lampson, W. (1973). "A note on the Confinement Problem.Communications of the ACM." (16(10)): 613-615.

Lipner, S. B. (1975). "A note on the Confinement Problem." Operating Systems Review, 9(5): 192-196.

McHugh, J. (1995). Covert channel analysis. I. H. f. t. C. S. C. o. T. Systems, Naval Research Laboratory.

Microsoft Corporation (2000). http://www.microsoft.com/resources/documentation/windows/2000/server/reskit/en-us/tcpip/part4/tcpappb.mspx.

Proctor, N. E. and P. G. Neumann (1992). "Architectural implications of Covert Channels." 15th National Computer Security Conference **Proceedings of the 15th National Computer Security Conference**: 28-43.

Route (1996). "Project Loki: ICMP Tunnelling." Phrack Magazine **7(49)**.

Rowland, C. H. (1996). "Covert Channels in the TCP/IP Protocol Suite."

V.Smirnov, V. (2003). "Firewall for Windows 9x/ME/NT/2000/XP 2003." NT Kernel Resources.

Wolf, M. (1989). "Covert channels in LAN protocols." LANSEC'89.

Authors Biography:

[A] Dr. William Buchanan …

WWW: http://www.dcs.napier.ac.uk/~bill
Email: w.buchanan@napier.ac.uk

[B] David Llamas is student of BSc (Hons) Software Technology at Napier University. His research area of interest is Information Hiding, focused on network security by the use of covert channels, and focused on hiding communications by the use of steganographical techniques. He is the webmaster of the specialised website: http://www.steganography.org

WWW: http://www.dcs.napier.ac.uk/~01009322
Email: david@inchcolm.org

“There is no worst attack than the one that can not be identified”

David Llamas,
Security researcher.