Deception on the network: thinking differently about covert channels

Maarten Van Horenbeeck
School of Computer and Information Science
Edith Cowan University
mvanhore@student.ecu.edu.au

Abstract

The concept of covert channels has been visited frequently by academia in a quest to analyse their occurrence and prevention in trusted systems. This has lead to a wide variety of approaches being developed to prevent and identify such channels and implement applicable countermeasures. However, little of this research has actually trickled down into the field of operational security management and risk analysis. Quite recently a number of covert channels and enabling tools have appeared that did have a significant impact on the operational security of organizations. This paper identifies a number of those channels and shows the relative ease with which new ones can be devised. It identifies how risk management processes do not take this upcoming threat into account and suggests where improvements would be helpful.

Keywords

Covert channels, deception, information assurance, risk management, threat modelling

INTRODUCTION TO COVERT CHANNELS

Covert channels have always proven useful in warfare. One of the first instances of their use can be found in Homer's Odyssey: "What a thing was this, too, which that mighty man wrought and endured in the carven horse, wherein all we chiefs of the Argives were sitting, bearing to the Trojans death and fate!" (Murray, 1919). After their decision to bring the Trojan horse inside, the Trojans' fate was forever sealed.

In computer science, many different authors have assigned even more different definitions to covert channels. From a trusted systems perspective, the most comprehensive definition to date is to be found in the 1993 'A guide to understanding Covert Channel Analysis of Trusted Systems', published by the US National Computer Security Center. This document, part of the TCSEC Rainbow Series, provides guidance on how to identify, prevent and audit covert channels (NCSC, 1993). It was the first step in systematically countering this modern form of deception.

The definition used in the document, while comprehensive, applies only to trusted systems using a nondiscretionary security policy. It defines a covert channel as: "Given a nondiscretionary (e.g., mandatory) security policy model M and its interpretation I(M) in an operating system, any potential communication between two subjects I(Sh) and I(Si) of I(M) is covert if and only if any communication between the corresponding subjects Sh and Si of the model M is illegal in M." (Tsai et al. cited in NCSC, 1993, p.9).

This model is applicable in many different situations, but commercially available computer systems and networks are usually less strictly regulated. Data transmitted on such networks may however still be relatively sensitive. As such, there has been a significant increase in the availability of tools employed to inspect outbound e-mail, web traffic and other network connections, to prevent or at least identify leaks of intellectual property.

At this time, leaks originating with individuals discussing proprietary information outside their employment context, or with those who copy information onto a thumb drive may still be more common, though it can be expected that as measures are taken to prevent them, computer based data smuggling will become more popular. A wide variety of tools and techniques enabling this are already available.

For the purpose of this paper, we will apply a simplified definition, more applicable to today's networked organizations. A covert channel is considered a 'transmission channel that may be used to transfer data in a manner that violates security policy'. (ISO, 1998)

In addition, the following taxonomy is proposed that further differentiates these channels by objective:

- Data smuggling: Covert channels can occur that are exploited with as primary goal to smuggle data into or out of a computer network, in which this data is not supposed to leave or enter that network according to the organizational security policy.

- Automated code execution: Covert channels can occur and be exploited with as primary goal to allow for the transfer into, and execution of code within the recipient system that should not normally be executed by the organizational security policy.

Initially developed as data smuggling tools, there has been a significant recent increase in the appearance of tools and malware of the second category, commonly identified as 'Trojan horses' or 'blended threats'. It is however important to note that automated code execution does not encompass all forms of Trojan horses. An executable file, sent through e-mail, willingly executed by the end user does not indicate the use of a covert channel: it is only when such file is trafficked in such way that it is not recognized as an "executable stream" that truly a covert channel is being applied.

CONTEMPORARY USE OF COVERT CHANNELS

IP Header Tunneling

IP Header Tunnelling was one of the first instances of network level covert channels. Each internet protocol packet contains a number of protocol headers that are used for a variety of purposes such as Quality of Service and Fragmentation. With this variety comes the risk of them being used to traffic data instead of managing network flows. By compressing data to a form that can be embedded in one such header, data can be transferred between networks covertly. While the headers are open to inspection, the embedded value is legitimate and not considered malicious.

An example is the IPID header field. This field provides for an identification number when network fragmentation takes place. The destination host knows that packets with the same ID refer to the same original datagram, which aids in re-establishment of the complete datagram.

Operating systems differ in the way they generate this IPID, but in general it is increased by one upon each packet transmitted. However, nothing is to stop us from actually crafting packets that use a predetermined value. By encoding data in separate 16 bit values and using them to fill the IPID field, then decoding them on the other end, a covert channel can be created (Rowland, 1995).

Naturally these types of techniques can be blocked by making sure that all connections are either proxied through another network device that rewrites these headers, or blocked on a perimeter firewall. As such, these techniques can easily be stopped by applying the well known technique of compartmentalization.

DNS Tunneling

The DNS protocol underlies the conversion of hostnames into IP addresses on the internet, and as such is a requirement for most computer networks. As it is a bidirectional protocol which makes it possible to send requests out onto the internet using an internal server, it has recently become popular as a tunnelling channel.

This technique is often used to make free use of subscription wireless hotspots: often these allow DNS resolution prior to requiring authentication. The user is successfully able to resolve external websites, after which all HTTP requests are rewritten by a transparent proxy to a local portal where the user authenticates and fulfils payment. While this system stops most people from browsing without purchasing a subscription, advanced users are able to tunnel other protocols, such as HTTP, through DNS to an outside server under their control. The concept of DNS tunnelling was initially identified by Oskar Pearson, in a posting to the Bugtraq mailing list in April 1998 (Pearson, 1998). It was later brought to fruition by Dan Kaminsky who wrote a number of tools that greatly extended the features of this covert channel (Kaminsky, 2004).

When using DNS tunnelling, an adversary sets up a server on the internet which is made authorative for a certain DNS zone. This means that the server is responsible for answering all DNS requests for, e.g. the zone "example.com". From a machine internal to the protected network, which does not have public internet access, the adversary then launches DNS requests for a random hostname, containing the information to be tunnelled out, within this zone. For example, if he wishes to tunnel out the information 'creditcard3506-3084-1028-4805vcc506', he would perform a DNS request for 'creditcard3506-3084-1028-4805vcc506.example.com'. This request is launched to an internal DNS server, who would relay it, no questions asked, to the adversaries' server.

Moreover, the external server can also respond to such request, perhaps with information to be tunnelled back into the protected network. By using encryption and encoding techniques on this data, large amounts of information can easily be tunnelled in and out, or connections could be established successfully.

As these are all legitimate connections according to the DNS protocol, there is very little detection that can alleviate this threat. Specific DNS tunnelling tools may use fixed protocols that can be detected using IDS

systems, but these protocols are easily altered. As such, signature based intrusion detection, in which a sensor inspects this traffic and tries to identify the tunnel being established, is not very effective.

Detection could be established by using statistical anomaly detection. In general, when data is transferred into or out of the network, there is significantly more DNS traffic between a set of hosts than is associated with regular DNS transactions. However, it would still be difficult to be absolutely certain: some benign services are much more likely to trigger false positives. In addition, DNS traffic is so high in volume that logging and evaluating each of the requests would pose a significant burden on any monitoring solution.

There is however one fairly accurate method of blocking this traffic. By completely disallowing internal hosts to perform DNS resolution, and forcing them to use a proxy server, the response to a DNS query would generally not make it back directly to the internal client. In addition, the outbound request is usually logged by the proxy, allowing forensic investigation and establishment of an audit trail.

HTTP Entity Tag Tunneling

While both of the above channels are still popular, countermeasures have been developed that allow an organization to, up to a reasonable degree of certainty, prevent their exploitation. However, it is clear that in large organizations, a significant number of other protocols are passed in and out of the network in order to meet business requirements. The technique we will discuss now, HTTP Entity Tag tunnelling, was developed by the author during a penetration test of a more sensitive network, in which significant action had been taken to prevent internal data from leaving the network. For example, there were checks on electronic devices being taken into and out of the computer processing centre and both the internal and external networks were segregated by firewalls, causing IP and TCP headers to be rewritten. Textual matching and verification of outbound e-mails was conducted, in addition to the countermeasures against DNS tunnelling mentioned above.

Only one protocol was allowed from the internal network to the internet, being the HTTP protocol required for web browsing. All this traffic was however passed through a proxy server to allow for antivirus and some content scanning. Most likely the common HTTP tunnelling technique would have been successful, being the trafficking of data in the actual HTTP request string. However, it has two disadvantages: when using a stock tool, it would have used any of a number of protocols. ATIS (2001) defines a protocol as being “a formal set of conventions governing the format and control of interaction among communicating functional units”. The adoption of any convention allows for easier identification thus leading to increased chance of our activities being discovered.

In addition, it was quite likely that the actual request headers were being logged by the proxy. One of the goals of this penetration test was to prevent the operators from establishing exactly which data had left the network, which would have been violated by the use of such tool.

The target data to be trafficked out was an Excel spreadsheet that contained a set of sensitive numbers. As HTTP was the only protocol available to transfer data out, a number of connections were made through the proxy using a variety of headers permissible under the HTTP RFC. By identifying which headers were passed through to the outside without alteration, a number of candidates were identified that could be used as a covert channel.

The ‘Content-MD5’ header was a prime candidate. When used, it is sent by a web server to a client containing the MD5 hash of the page being served (Fielding et al, 1999). This hash can be used by the client to identify whether the page was changed as compared to the version currently in its cache. The MD5 hash algorithm produces 128 bits of output, and as such, it would be possible for us to encode the data to be transferred, split it into 128 bit pieces and concatenate these again on the destination host.

Limitations applied to the use of this header, however: the fact that only 128 bit blocks could be used would significantly limit the bandwidth allowed. In addition, while less applicable to our situation, the transfer would only have allowed for one-way communication.

One set of bidirectional headers was investigated further: ‘ETag’ & ‘If-None-Match’. Entity tags in HTTP allow a client to verify whether its locally cached copy is still current. When a specific document is served, the web server is allowed to include an ‘ETag’ header that contains a string which describes the page. Upon first retrieval, the client caches both the page and its accompanying ETag. When the client visits that resource again, it is allowed to send an ‘If-None-Match’ header to the server, displaying the list of ETags it currently has cached (Fielding et al, 1999).

If the current version on the server matches, the server may respond with a message stating that the page has not been updated. Should however, none of the ETags match, the updated version of the page is sent. Interesting here is that while the HTTP/1.1 RFC describes how ETags should be implemented, it does not discuss exactly how they are calculated or what they should look like. This means that we can choose their size without violating the protocol specifications, and if we use this string to traffic other types of information, such as our binary Excel file, have a significant amount of bandwidth at our disposal.

A tool, Wondjina, was developed and proved to be successful in trafficking this file out of the network. While the technique employed was not invisible to the proxy, the exact data moved out of the network could not be identified through analysis of the proxy server logs. This tool has been tested with a range of proxy servers and has proven to be quite effective.

Some potential countermeasures were investigated:

- In theory it would be possible for perimeter proxies to perform stateful checking of entity tags. The first such tag in an HTTP connection is however quite likely not to match and would trigger a false positive. An algorithm could allow for such first occurrence, thereby making the technique obsolete by constraining its bandwidth. Nevertheless, an issue of significance was identified that would prevent such methodology: some web servers use a server specific file value, the inode, to calculate the ETag. Identical requests for a file hosted on a web cluster could result in multiple different entity tags. This would cause an unacceptable amount of false positives.
- Verifying entity tag constitution: Apache, a common web server, produces an entity tag which is similar in constitution at all times. By detecting tags that would not match this profile, potential tunnelling could be identified. There are however two limitations: our adversary could produce identical entity tags, which would trigger no suspicion but significantly limit his bandwidth. Also, not all web servers may create tags in an identical way, causing false positives.
- Completely dropping all usage of entity tags inbound or outbound on the perimeter proxies. Alternatively, by forcing HTTP/1.0 the mechanism could also be disabled (as only HTTP/1.1 supports its use).

There are a significant number of disadvantages involved in this specific technique that are beyond the scope of this paper. Nevertheless, it does provide a good example of how tunnelling techniques can be devised easily and at low cost to match specific security situations.

Covert encoding or steganography

Each of the above techniques specifically provides a way to piggyback information onto an existing protocol. An even more advanced way of tunnelling information is by covertly encoding it in benign data, legitimately transferred over existing protocols. Dubbed steganography, a term derived from Greek meaning covered writing, these techniques consist of altering bytes in predominantly lossy protocols, which do not lead to a perceivable change in data quality but do allow information to be embedded without being identified.

While research has identified ways to include information in benign items such as digital certificates, steganography is most often identified with the common JPEG and MP3 formats. Research was conducted by Niels Provos to evaluate how extensively this is being used in real-life, by mathematically evaluating over one million graphics files from Usenet as well as 2 million images from eBay auctions. None of these was found to contain hidden information (Provos, 2002).

While their evaluation techniques did not intend 100% accuracy – only output by known steganographical tools could be detected – they do give the indication that most likely, real-world use of these techniques is either limited in scope, or used in cooperation with additional security techniques such as restricting distribution of the end product to private e-mail communication.

ONE STEP FURTHER: BLENDED THREATS

While the above techniques have proven their use predominantly in disseminating information that should not leave or enter an organization, and as such pose a significant threat by themselves, they are still no match for the vicious creature the illustrious Trojans allowed into their midst. ‘Trojan horse’ software has been known in data processing fore quite some time, as software which purports to perform a certain function while actually completing a less desired act upon execution. In general, an end user was still required to manually ‘open’ the horse, by clicking on, for example, an e-mail attachment. Certain Trojan horses can now however be considered covert channels under our previous definition: they combine an illicit transmission channel with automated execution. Some of today’s application level vulnerabilities serve as an example.

In November 2004, the equivalent of a ‘Trojan horse on steroids’ appeared on the internet. A vulnerability was identified in the Microsoft GDI+ or Graphics Device Interface library, more specifically in its ability to interpret and display JPEG image files (Microsoft, 2004). JPEG is one of the most common formats to distribute images on the internet and is used by the majority of today’s websites.

This specific vulnerability consisted of a software flaw, a so-called buffer overrun, in the code that was used to interpret and display such images. As this flaw was located in the base operating system libraries, every single application that used these GDI+ libraries, including most internet browsers, editors and graphics applications, was vulnerable to this flaw.

The impact was quite significant. An attacker could modify existing images to contain code to exploit this specific vulnerability, and as such execute arbitrary code on any affected operating system by having an end user merely view the image file. This differed from previous vulnerabilities, where the end user in general had to run a specific application, or at least click on an item in order to have a malicious act occur. In this case, merely opening a webpage which had a malicious image embedded proved sufficient.

For years, Information Security professionals had focused on creating awareness amongst end users regarding the security repercussions of double clicking on executable files. This vulnerability however upped the ante; users were no longer safe even while browsing seemingly benign web sites. Shortly after, similar vulnerabilities were identified in the processing of Windows Meta File documents (Microsoft, 2005).

These vulnerabilities had significant repercussions for threat mitigation purposes. Signature based matching; both in anti-virus and network IDS systems did not prove up to this specific threat. While signatures were written for the payload of the initial exploit, later exploits used more buffering space between the different payload components. This caused later releases of the exploits not to be detected (ISC, 2005). In addition, larger parts of TCP streams required inspection instead of single packets, which proved difficult for the then current IDS systems (ISC, 2006).

While perimeter proxies and e-mail inspection engines could have countered this vulnerability, they were usually configured to only inspect executable content. Images, considered for passive display only, were not inspected by default. In addition, when JPEG scanning was enabled for the above, there were no generic signatures available that looked for malicious code inside these images. JPEG and WMF files were never considered a viable attack vector. While heuristic analysis – the inspection of files for generic malicious functions instead of specific signatures - may have been available for executable files, this was not the case for JPEG files.

In addition, some anti-virus engines initially detected JPEG image files by their file extension. Such installation would have rendered them useless. The Windows GDI+ uses internal file structures to identify a JPEG file, and any renamed file would as such have passed through the engine without inspection. As such, even when JPEG scanning was enabled, it was not effective in all cases.

Many of the above items posed significant issues during the initial vulnerability outbreak. While most organizations have now put measures in place to remedy these and anti virus vendors have released new signatures that are able to counter these threats – there were obviously some structural issues here. One of them was a lack of imagination: how could images be executable?

COVERT CHANNELS: A PARADIGM SHIFT

Covert channel analysis is a long-standing academic discipline. In general, such channels are grouped by the shared resource that allows for communication between two entities. Two common shared resources are ‘timing’ and ‘storage’ channels. Timing channels appear when a process can influence the response time of the system to another process. Storage channels on the other hand, reflect the use of a shared storage medium to exchange information (Cabuk, Brodley & Shields, 2004).

The before mentioned ‘A guide to understanding Covert Channel Analysis’ was the first hands-on document to assist in covert channel analysis in software development. It summarized the state of the art in 1993, and was developed to assist developers and evaluators of systems that required B2 or higher accreditation under the Trusted Computer System Evaluation Criteria or TCSEC.

While this paradigm of covert channel research has proven very useful in the development of high security systems, it applies less to the types of systems predominantly used by commercial organizations. The only commonly used commercial operating system assessed under the TCSEC, Windows NT, was accredited only to the C2 level, which excludes covert channel analysis.

In addition, covert channel analysis is usually only applied to individual system components, such as an operating system. However, in today’s network environment, such operating system would be deployed within a computer network, and different criteria would apply to define whether covert channels exist that pose a threat to the information carried by the network. Covert channels would not so much be an issue on a single server, but their existence between the inside and outside, untrusted network, would be a risk factor.

As contemporary covert channel analysis focuses on single channels, it measures the viability of such channels in their capacity or bandwidth. In 1987, the NCSC stated that covert channels exceeding more than ten bits per second need to be audited by the Trusted Computing Base, or TCB, of a system (NCSC, 1987). From the viewpoint of an information security risk manager, however, bandwidth may not be a prime concern in itself.

When present in a network, a single covert channel could lead to multiple threats materializing:

- DNS or HTTP tunnelling could be abused by employees to connect to services out on the internet which they should normally not be allowed to connect to. Merely one example is the use of DNS tunnelling to access, from a corporate machine, a web mail application. This type of easy, out-of-the-box tunnelling method would easily allow a user to bypass most other components that are implemented to protect against e-mail borne virus threats or any e-mail inspection that may be in place. These types of web applications could easily be filtered on a web proxy, but when they are tunnelled over DNS, such filters are rendered useless;
- Code could be transferred to an internal client that, once inside, exploits internal vulnerabilities that are usually not exposed to the internet;
- Intellectual property could be trafficked out of an organization.

While the original measurements of available bandwidth are still useful in some of these situations, the above shows there is an obvious need to not only review covert channels in individual systems, but also to design methodologies to test for them in integrated systems such as computer networks. Such methodology would need to review the information handling infrastructure holistically and integrate in existing risk management methodology. In addition, each threat presented above poses unique bandwidth requirements. One catch-all metric to assess their importance is no longer sufficient.

RISK MANAGEMENT APPROACHES TO DECEPTION

Covert channels perhaps best compare to the physical act of smuggling. Risk management approaches to smuggling are often based on probabilistic risk management, the measurement of the probability of an incident taking place based on intelligence gathered on the risk level, our current system state and our adversaries. This has for example been investigated in the field of Nuclear Material trafficking (Scott, 2002). Based on such probability analysis, the amount of uncertainty related to police decision making is reduced. While a valid and proven methodology, it does not completely transfer to the online world.

Covert channels actually do correspond, as many modern cyber attacks with a general societal change towards asymmetric threats: threats that take place in an unexpected form and do not require large troop movements to succeed. This makes them difficult to predict, and dependant on either sudden insight or carefully acquired intelligence. Covert channels, however, carry some additional complexity.

In themselves, they rarely constitute an attack in themselves: they are generally part of a larger incident such as a compromise. Gathering intelligence on adversaries is also quite difficult: in general, an organization will only be able to gather intelligence on its own operations, such as network and security documentation. Acquiring information on detailed threats and even on the latent threat level is quite difficult, given the fact that most organizations are often reluctant to report on security incidents (CSI, 2006). In addition, the nature of covert channels makes that they are generally not detected unless they lead to further incidents.

This combination of issues makes the use of probabilistic risk management purely for the assessment of covert channels very difficult. It impacts the ability of an organization to plan efficiently for the mitigation of such threats.

Security management standards, such as ISO17799 or ISO27001 do not deal with covert channels directly, but assume them to be dealt with under broad network segregation and network connection controls. These controls are then puzzled in using policies and procedures – in a technical sense tools such as the ‘Open Source Security Testing Methodology Manual’ or OSSTMM are quite popular (ISECOM, 2006). In such documentation, however, there is little representation of covert channel analysis.

Lacking is an additional form of testing which is already commonly applied in the development of network components or software applications: threat modelling. In general, when assessing their security posture, most organizations use a number of tools:

Continuous risk analysis efforts such as vulnerability assessments concentrate on vulnerabilities in specific software components. While they may also perform some limited testing of firewalling or intrusion prevention capabilities, they do not focus specifically on generic covert channel capabilities.

Acceptance tests such as penetration testing could be useful in the identification of covert channels. Unfortunately, they are usually interested in finding 'one way of many' to compromise a system. They do not specialize in identifying each of the potential vulnerabilities. The types of threats posed by covert channels are so drastically different from immediate compromise that they may be disregarded by any penetration test not specifically designed for that purpose.

During the systems development lifecycle, mainly within software development, an additional tool, 'Threat modelling' is also quite popular (Casteran, 2006). It "involves understanding the complexity of the system and identifying all possible threats to the system, regardless of whether or not they can be exploited" (Myagmar, Lee & Yurcik, 2005).

By nature it is open-ended: security requirements are defined. Based on these requirements an exhaustive listing can be made of all threats that apply. One of these threats could be the illicit transferring of data from trusted, internal networks, to the internet. The information system, as implemented, can then be reviewed in detail specifically based on these threats. While a detailed technical testing framework needs to be developed, threat modelling can be considered a candidate process.

CONCLUSION

Millen (1999, p. 114) stated "the one question that everyone asks about covert channels is whether they are a real threat". This paper diversifies that question and identifies three separate threats that can materialize when a covert channel exists within a network. As long as there are sufficient 'open channels' through which confidential information can leak from an organization, the capability to transfer data from a trusted network to an untrusted one may not be a significant issue. However, with increased measures being implemented to counter the more obvious issues – such as the free transfer of USB sticks in- and out of an organization, use of computer based covert channels is certain to rise. In addition to this threat, it is important to realize that covert channels could also lead to direct circumvention of carefully crafted filtering systems, or be a stepping stone to compromise through 'blended threats'.

This paper presented a case for the integration of covert channel analysis into information security risk management programs. It describes a number of popular covert tunnelling methods that have been used in the past and for which stock tools are available. In addition, it also introduces the reader to one new channel to identify the ease with which channels can be identified for very specific security situations.

The current tools being used to conduct risk assessment and analysis within an organization are not sufficient to identify the risk of covert channels. It would be useful to develop a framework that allows for the holistic identification of covert channels in information networks. This should be integrated in existing risk management methodology, and allow information security managers to obtain a more accurate view of the threat, improving security decision making. The tool of threat modelling is briefly touched upon and identified as a candidate for this purpose.

REFERENCES:

ATIS (2001) *ATIS Telecom Glossary 2000*. Washington: Alliance for Telecommunications Industry Solutions. [Online] Available at: http://www.atis.org/tg2k/_protocol.html

Cabuk, S., Brodley, C. E. & Shields, C. (2004) IP Covert Timing Channels: Design and detection. *Proceedings of the 11th ACM conference on Computer and communications security*, pp. 178-187. Washington: ACM Press.

Casteran, F. (2006) Risk Management through Threat Modelling. *CIO Update.* [Online] Available at: http://www.cioupdate.com/trends/article.php/3622776

CSI (2006) *CSI/FBI Computer Crime and Security Survey 2006.* Computer Security Institute. [Online] Available at: http://www.gocsi.com

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T. (1999) *RFC 2616: Hypertext transfer protocol*. Reston: The Internet Society. [Online] Available at: http://www.w3.org/Protocols/rfc2616/rfc2616.html

ISC (2006) *New exploit released for the WMF vulnerability*. Internet Storm Center. [Online] Available at: http://isc.sans.org/diary.php?storyid=992

ISC (2005) *More WMF Signatures*. Internet Storm Center. [Online] Available at: http://isc.sans.org/diary.php?storyid=980

ISO (1986) *Information technology. Vocabulary. Control, integrity and security*. Geneva: International Organization for Standardization

ISECOM (2006) *OSSTMM Open Source Security Testing Methodology Manual*. [Online] Available at: http://www.isecom.org/osstmm/

Kaminsky, D. (2004) *DNS Tunneling presentation*. [Online] Available at: http://www.doxpara.com/bo2004.ppt

Microsoft (2004) *Microsoft Security Bulletin MS04-028: Buffer overrun in JPEG Processing*. Redmond: Microsoft Corporation. [Online] Available at: http://www.microsoft.com/technet/security/bulletin/MS04-028.mspx

Microsoft (2005) *Microsoft Security Advisory (912840): Vulnerability in Graphics Rendering Engine.* Redmond: Microsoft Corporation. [Online] Available at: http://www.microsoft.com/technet/security/advisory/912840.mspx

Millen (1999) 20 years of covert channel modelling and analysis. *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pp. 113-114. Oakland: IEEE.

Murray (1919) *Homer. The Odyssey*. (Translation by) Cambridge: Harvard University Press.

Myagmar, S., Lee, A. J. & Yurcik, W. (2005) Threat modelling as a basis for security requirements. *Symposium on Requirements Engineering for Information Security*. [Online] Available at: http://www.projects.ncassr.org/threatmodelling/sreis05.pdf

NCSC (1993) *A guide to understanding covert channel analysis.* Fort Meade: NCSC

NCSC (1987) *A Guide to Understanding Audit in Trusted systems*. Fort Meade: NCSC

Pearson, O. (1998) *DNS Tunnel – through bastion hosts*. Bugtraq posting. [Online] Available at: http://seclists.org/bugtraq/1998/Apr/0079.html

Provos, N. & Honeyman, P. (2001) Detecting Steganographic Content on the Internet. *Proceedings of the Network and Distributed System Security Symposium – San Diego 2002*. Reston: The Internet Society.

Rowland, C. H. (1995) Covert Channels in the TCP/IP Protocol Suite. *First Monday: Peer-reviewed journal on the Internet*. Chicago: University of Illinois. [Online] Available at: http://www.firstmonday.org/issues/issue2_5/rowland/index.html

Scott, B. (2002) Decision-based model development for nuclear material, theft, smuggling and, illicit use. *Proceedings of international conference on physical protection (NUMAT)*. Salzburg: University of Salzburg. [Online] Available at: http://www.numat.at/list%20of%20papers/scott.pdf

COPYRIGHT

Maarten Van Horenbeeck ©2006. The author/s assign SCISSEC & Edith Cowan University a non-exclusive license to use this document for personal use provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive license to SCISSEC & ECU to publish this document in full in the Conference Proceedings. Such documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Web. Any other usage is prohibited without the express permission of the authors