Intruder Detection System (IDS) Evasion

Solahuddin Yusuf Fadhlullah
Mohamad Adha Mohamad Idin
Mohd Halim Mohd Noor

ABSTRACT

Intrusion detection system (IDS) is a technology, be it a software or hardware, that automatically scans events occurring in computer or network for signs of security problems. IDS evasion is the act of executing any illegal activity towards the target host without being detected by the IDS. Such activities can be invading, hacking, transmitting malicious data or simply inflicting damage. The methods used in the evasion may be insertion, evasion and ambiguities. These attacks can be conducted in any layer in the OSI model. Thus, in this study, major and well known evasion techniques are exposed and discussed. Countermeasures are also mentioned and listed down in order to mitigate the threat of IDS evasion.

Keywords: *intruder detection system, hacking, IDS vulnerabilities, IDS evasion countermeasures.*

Introduction

Intrusion detection systems are the 'burglar alarms' or rather the 'intrusion alarms' of the computer security field (Axelsson, 2000). It should be noted that IDS is a part of the security procedure that is usually established by an organization which complements with other security services such as firewall and anti-virus. There are two types of IDS which are host-based IDS and network-based IDS. However, both approaches have their own advantages and disadvantages as discussed by Basicevic, Popovic, and Kovacevic (2005). Furthermore, testing and comparing different

ISSN 1675-7939

IDS, be it host or network-based, is difficult as each IDS has different operational environment and event stream which adds complexity in determining the best security solution (Mutz, Vigna, and Kemmerer, 2003). Vulnerabilities, holes and gaps within IDS make it possible to craft a complete IDS evasion technique. Even though current modern IDS are considered reliable, still, hackers are not lagging much to keep up with the challenge to get the upper hand against security vendors. Each IDS in the market today has their strengths and weaknesses – there is no perfect IDS out there. Plus, the architecture of computer systems and the need for connectivity will always open doors for intrusion.

The first principle of evasion technique is to know the IDS system; whether it is a host-based or network based IDS or both altogether at the same time. If we can figure out the types and specific version of IDS deployed, we can directly seek out its known vulnerabilities. The second principle is to know the network that we are targeting. It will be much easier if we know the placement of the IDS and the topology of the network itself. The last principle is to be familiar with the target host in order to abuse target-specific behaviours (Caswell & Moore, 2006). Finally, attackers also exploit the lack of knowledge and the weaknesses of IDS implementation in terms of configuration, version and protected communications protocol stack (Basicevic, Popovic & Kovacevic, 2005).

Overview of Fundamental IDS Evasion Techniques

We can drilldown the basic fundamentals used in the attack against IDS into four categories which are insertion, evasion and ambiguities. Another possible way of classifying the attack is based on the OSI model where hardware evasion occurs at layer 1 (physical) and 2 (data link), traffic evasion is executed in layer 3 (network) and 4 (transport) while the software evasion is aimed at layer 5 (application).

Insertion

Attackers will deliberately 'insert' additional information into the packet and send it to the target host. This technique involves inserting 'bad' data which will be utterly rejected by the target host but accepted by the IDS. In order for this to work, the attacker hopes to trick the IDS by making the IDS believe that all portion of the packet was received by the target

host when actually a part of it was discarded by the host. This is meant to evade intrusion detection by crafting a different signature and pattern of attack. For example, the keyword of the real attack is **INTRUDE**. The attacker will insert additional data to the keyword to disguise as another type of word to evade detection. The attacker could modify the original keyword to be '**INTR**O CLUB **DE**SK'. Here, the attacker adds the letters O,C,L,B,S and K so that the IDS can no longer picks up the original specific word **INTRUDE**. However, in turn, those additional letters are invalid inputs to the host and are rejected. Hence, the only input letters received by the host are actually INTRUDE. This marks a successful IDS evasion. In short, this kind of attack hopes that the end system and the IDS would construct different outputs.

Evasion

Evasion technique is quite the opposite of the insertion method. Instead of adding more information into the packet, evasion technique aims to make the IDS misses part of the malicious packet. An IDS that mistakenly rejects such a packet misses its contents entirely (Ptacek & Newsham, 1998). Thus far, there is no newer evasion procedure published since Ptacek & Newsham (1998) publicized their work (Pastrana, Orfila, and Ribagorda, 2011). This works by ensuring that the end-system accepts a valid packet that contains the whole malicious content but is partly missed by the IDS. When the IDS tries to match for a known pattern, it will not be able to do so as portion of the attack content is missing. For example, if the attack packet contains the content of **INTRUDE**, it will be fully received by the host. However, due to evasion technique, the packet obtained by the IDS is only **INRUDE**, where the letter T is missing. Thus, this shows another successful IDS evasion. Evasion type of attack is the easiest to be exploited and proves to be disastrous to the accuracy of an IDS.

Ambiguities

IDS is susceptible to ambiguity problems. For example, the IDS cannot determine whether the target host will accept or reject a packet. Furthermore, due to protocols, restrictions and special cases, the target host might accept what seems to be an outright rejected packet by the IDS and vice versa. This creates problems for the IDS to discover any intrusion as attackers know how to exploit this situation. Two cases in

Ethernet ambiguity are when there exist slightly oversized frames and multiple VLAN headers (Caswell & Moore, 2006).

OSI Layers Attack

Layer 2 attacks will not be covered in this paper due to its impracticality. Attacking layer 2 is quite useless because it requires local media access and IDS would most likely drop the attacking frames if layer 2 portion is being tampered. However, both layer 3 (network) and layer 4 (transport) are an open playfield for attackers to attempt IDS evasion. Layer 5 (application) is also a favourite target for intruders. These layers will be discussed in more detail in the next part of this study.

Known Evasion Techniques

Network and Transport layer

i. IP fragmentation

Fragmentation is the method of splitting IP packets into smaller packets. These smaller packets will be transmitted and then reassembled at the destination (Timm, 2002). The minimum fragment size possible is 8 bytes. There are two methods commonly used to elude IDS. The first method overwrites a section of a previous fragment while the second method overwrites a complete fragment (VeriSign team, 2006). Here, we will look into two examples of fragmentation attack as exposed by the VeriSign team.

Attack 1: Overlap Method
Packet 1: GET /cgi-bin/
Packet 2: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/../**ph**xx
Packet 3: **f?**

This example of fragmentation can overwrite the 'xx' portion of Packet 2 with the data in Packet 3, making the overall information resembles the following:

GET /cgi-bin/ aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/../**phf?**

Attack 2: Overwrite Method
Packet 1: GET /cgi-bin/

Packet 2: normal_filename.cgi
Packet 3: /aaa/../aaa/../aaa/../**ph**xx
Packet 4: **f?**

This example is similar to the first method, but here, the <xx> portion is overwritten and the 'normal_filename.cgi' packet is completely replaced with the last two packets. This leaves GET/cgi-bin/phf? as the end result. This PHF command is a very old trick which exploits a vulnerability in the 'phf cgi' script within the HTTP 'GET' request. PHF is a program that usually comes pre-installed on every UNIX machine which allows users to download any file from the server, including the password file (IWS, 2001). However, this 'cgi' weakness has already been removed from Web server distributions (Raffy, 2004).

ii. Time-to-live attack

The purpose of time-to-live (TTL) attack is to deliberately send garbage data that will only be accepted by the IDS but not by the target host. This enables the attacker to insert useless data into the IDS stream processing in order to render the IDS incapable of matching any known attack signature. Using small TTL flag allows attackers to send packets addressed to the end host to reach the IDS without the packet ever getting to the end host (Raffy, 2004). Whereby sending packets with a large TTL flag will guarantee the packet to reach the end host. However, to carry out this attack, the attacker must first know the internal network topology. IDS evasion is possible only when the attacker knows the exact distance to the target host and the IDS. An example of the same PHF attack as mentioned by VeriSign team (2006) is shown below.

Packet 1: GET /cgi-bin/**p** TTL 10
Packet 2: some_file.cgi?= TTL 5
Packet 3: **hf?** TTL 10

This example assumes that the end host is located at 10 TTL limit and will receive the data. It is also assumed that the IDS is within the 5 TTL limit. This means that the end host will receive "GET /cgi-bin/phf?" while the IDS receives "GET /cgi-bin/psome_file. cgi?=hf?".

iii. Unicode exploits

Unicode is a character representation that gives each character a unique identifier for each written language to uniformly facilitate the computer representation of each language (VeriSign team, 2006). The problem here lies where it is possible to have multiple representations of a single character. Unicode encoded vulnerability has also been detected in Microsoft Internet Information Services (IIS) 4.0/5.0 that allows attacker to view restricted files on the IIS server.

iv. Session splicing

This technique exploits how some IDS do not reconstruct sessions (multiple blocks of continuous packets) before performing pattern matching on the data (VeriSign team, 2006). Reassembling session requires a lot of resources and puts burden on the processor. Thus, some IDS only reassembles parts of a session and this is where the attack is focused at.

Session splicing is done by dividing into several packets and ensuring that no single packet matches any patterns within an IDS signature record. Given that the attacker knows the type of system used, he can add delays between packets to bypass reassembly inspection. If a packet is not received within a defined amount of time, many IDS will just stop reassembling the data stream while the application under attack still keeps the session open. The repercussion is that any packet or data from the session that the IDS stops reassembling will be susceptible to malicious activity sent by the attacker which would go unnoticed.

v. Invalid RST packets

This is a type of transport control protocol (TCP) exploit in the transport layer. In order for TCP to ensure that communication is reliable, it uses the checksums feature. Every transmitted segment has a checksum entry and it is checked at the receiver. If a checksum differs from the checksum expected by the receiving end, the packet will be dropped. The TCP can also end active communications by using an RST packet. Both of these features will be utilized by the attacker to craft the intrusion.

The invalid RST packet attack is carried out when the attacker sends RST packets with an invalid checksum, which in turn causes the IDS to stop processing the stream because the IDS thinks the

communication session has already ended. However, the end host still accepts the packet and verifies the checksum value, and then drops the packet if it is invalid. Some IDS might interpret this packet as an actual termination of the communication and stop reassembling the communication (Versign team, 2006). This will allow the attacker to continue his communication with the end host whilst the IDS assumes that the session is already terminated.

vi. Urgency flag

Urgency flag is another feature within the TCP protocol. It is used to mark data as urgent and an urgency pointer is used to point at the beginning of an urgent data within a packet. When the urgency flag is set, all data before the urgency pointer is ignored, and the data to which the urgency pointer points is processed (VeriSign team, 2006).

The flaw lies in the behaviour of some IDS that disregard the TCP urgency feature. The IDS will keep on reading and analysing garbage data that is inserted by the attacker before the urgency flag without the consideration for the end host's urgency flag handling. This means that the IDS will accumulate more data than the target host actually processed.

According to the 1122 RFC (Internet Engineering Task Force Request For Comment memorandum), the urgency pointer causes 1 byte of data next to the urgent data to be lost when urgent data is combined with normal data (VeriSign team, 2006). Below, we will see a scenario of an urgency flag attack to inject the message INTRUDED within the target host.

Packet 1: INT
Packet 2: RUD Urgency Pointer: 3
Packet 3: XED
End result at end host: INTRUDED (known attack signature)
End result at IDS: INTRUDXED (different signature, may be missed by the IDS)

vii. Polymorphic shellcode

A shellcode is a small piece of code used as the payload in the exploitation of a software vulnerability (Shellcode, n.d.). Polymorphic shellcode allows attackers to hide their shellcode by encrypting it in a simplistic form (VeriSign, 2006). As the word

polymorphic implies, the attacker can send shellcodes in completely different forms each time it is sent, thus, evading signature detection. Moreover, encryption will make it difficult for the IDS to identify the data as a shellcode.

Application Layer

There are millions of applications in our world today, which enable many different forms of evasion. IDS evasion is possible by exploiting protocol code differences, error condition handlings, vendor specific extensions and etc. There are even lots of tricks that can be played upon many protocols including HTTP, FTP, SMTP, DNS, SunRPC, DCERPC, SMC and the list goes on. Furthermore, many application that deals with media such as images, videos and audios implement some form of compression in order to reduce the file size and thus increasing the data transfer speed. These features combined with evasion techniques at network and transport layer enable attacks to craft sophisticated attack that can trick the IDS in the application layer.

i. Pattern-based (signature-matching) weaknesses

This is by far, one of the simplest and basic method used to elude IDS. A lot of evasion techniques are based on exploiting the pattern-based detection approach implemented by most IDS. Pattern-based detection uses pattern matching to identify potential attacks based on known vulnerabilities or commonly used strings within exploit code (VeriSign team, 2006). Pattern-based detection has lots of flaws, as even slight changes in input can bypass detection patterns. Another thing worth noting is that not all inputs need to be the same in order to trigger certain specific vulnerabilities. The following example shows a pattern that would be processed from a HTTP session and an altered unintelligible version that attempts to bypass the pattern as revealed by VeriSign team (2006).

Attack pattern:
GET /cgi-bin/phf?

Obfuscation:
GET /cgi-bin/aaaaaaaaaaaaaaaaaaaaaaaaa/..%25%2fp%68f?

Both versions result in the same output, yet look very different. This is another example of the GET/cgi-bin/phf? exploit as seen earlier.

ii. Buffer overflow

Attackers can launch a very effective attack by inflicting buffer overflow on the target host. This is because the technique used is based on the attacker's code of choice and also the fact that allowed protocols in default to go through firewalls without much obstruction. Attackers can also exploit mismanaged bound checks and craft more types of attack. In addition, this technique can evade both host and network based IDS (Ingeborn, 2001).

Buffer overflow exploitation is a rather advance technique and requires it's own separate study to explain everything in detail as it involves internal registers, programming and understanding the architecture of the target software. However, the basics in carrying out this method is to employ code insertion method after a successful abuse of mismanaged bound check.

As a summary, buffer overflow exploit is a flexible and smart attack as the attacker can continue to intrude and hack the system possibly undetected while using no new TCP connection or TCP ports. It can utilize pre-loaded functions and .dlls (Microsoft Windows Dynamic-link Library) by the target host, re-use socket descriptor and totally evading host based IDS as there are nothing for it to record in the event log. (Ingeborn, 2001).

iii. SMB

SMB is a transport protocol used for remote file access, system administration, network authentication and remote procedure calls (Caswell & Moore, 2006). Among SMB based vulnerabilities are 'malware' propagation, remote registry access and authentication attacks. Although SMB is a transport layer protocol, the method of attack is done on the application layer. The followings are two known SMB exploitation cases exposed by Caswell & Moore (2006):

a. Segmented read and write operations attack is independent of both IP and TCP layers. This attack can evade 'malware' and DCERPC signatures, while requiring the IDS to track all the data length and offset in order to detect the attack.
b. Data and parameter padding attack is carried out by filling them with bogus data. Attackers can insert fake SMB requests and at most only triggers low risk IDS signatures.

There are a lot more SMB attacks out there such as SMB transaction PIPE string exploit, SMB CreateAndX path names abuse, Unicode and non-Unicode exploits, request stacking and AndX chains. All of these techniques require intensive knowledge and understanding on multiple fields to be exploited by attackers.

iv. DCERPC exploits

DCE/RPC which stands for Distributed Computing Environment/ Remote Procedure Calls is a remote procedure call system that allows software to work across multiple computers, as if it were all working on the same computer (DCE/RPC, n.d.). It supports multiple protocol including TCP, HTTP, UDP and SMB. Hence, DCERPC offers a lot of feature such as fragmentation, multiple data representations and function. Below, we will look into a few evasion methods as explained by Caswell & Moore (2006).

a. DCERPC bind evasion is done by binding multiple Universally Unique Identifier (UUID)s at once. A UUID is an identifier standard used in software construction and has the function to enable distributed systems to uniquely identify information without significant central coordination (Universally Unique Identifier). Attackers can also elude IDS detection by binding to one UUID and then use the AlterContext function.
b. DCERPC Call evasion is implemented by encrypting data with packet privacy, fragmenting the data across many requests, append random data to Network Data Representation (NDR) stub and perpend an Object ID.
c. DCERPC transport layer evasion can be crafted by running RPC over HTTP via RpcProxy. By using the idempotent flag, the attacker can use one packet UDP function call to fashion his attack.
d. Ports and processes exploits may be carried out through pipe sharing by shared processes. Other types of IDS evasion methods are DCERPC – NDR strings, ISystemActivator, and ISystemActivator path exploits.

v. Unicode exploits

Unicode that is used in many applications and protocols can be misused to craft an intrusion. Furthermore, there are a lot of Unicode that can be implemented such as UTF-7, UTF-8, UTF-16LE and

others. This presents a lot of possibilities to invent an attack. For instance, Unicode can be exploited in HTTP and Object Oriented Programming (OOP) using UTF-16BE. Below is a simple example as brought forward by Caswell & Moore (2006).

Using UTF-8 overlong strings to encode letter 'A'.
Results: 41, c1a1, e081a1, f08081a1, f88008081a1, fc80808081a1

Here, we can represent an encoded letter 'A' in many forms and this technique can be utilized to evade known signatures.

vi. Javascript IDS evasion

JavaScript is a scripting language widely used for client-side web development (Javascript, n.d.). It is normally used for websites and also for enabling scripting access to objects embedded in other application. The following is an example to insert the word INTRUDE using Javascript by Caswell & Moore (2006).

<script>document.write("INTRUDE")</script>
<body onLoad=document.write(INTRUDE);>
document.write (unescape('%45%56%49%4C'));
<font style=background:url(javascript:document.write('INTRUDE'));>

vii. HTML based evasion

HTML stands for HyperText Markup Language and is used for web pages. It provides a means to describe the structure of text-based information in a document – by denoting certain text as links, headings, paragraphs, lists, and so on – and to supplement that text with interactive forms, embedded images, and other objects (HTML, n.d.). The example below shows the command to execute the same operation as shown in Javascript earlier (Caswell & Moore, 2006).

<OBJECT ID="INSERT" TYPE="text/html" DATA="data:text/html;base64,ZXZpbCB0ZXh0">INTRUDE</OBJECT>

viii. SSL exploits

Secure Sockets Layer (SSL), are cryptographic protocols that provide security and data integrity for communications over TCP/IP networks such as the Internet (Transport Layer Security). SSL has already been replaced with Transport Layer Security (TLS) but the technique utilized for IDS evasion is effective nonetheless which is by using

encryption. Smart attackers can compromise and hijack existing certificate and make their entry into the target host by convincing the end user to ignore warnings. This can be done by using SSL wrapped CGI proxy servers (Caswell & Moore, 2006).

ix. Legiment techniques

This is a relatively new technique proposed and published by Ajit Hatti (Ajit, 2007). 'Legiment' method focuses on exploiting application protocols to evade modern IDS. It requires the attacker to know the exact target application for the modus operandi to work. From here on, any existing exploit can be crafted to totally evade and trick the IDS. Ajit (2007) has shown four examples of MS03-046, an Exchange server exploit. All of these examples are launched using Telnet and the description are as discussed below.

The first example is by sending an invalid sender in <MAIL FROM> and keeps the Exchange in command state unless a valid sender is sent. Thus, IDS can't decide the validity of the sender. Using this method, IDS can be pushed in DATA state while keeping Exchange in command state. Thus, any commands going after it are ignored by IDS as a part of the message body.

The second example is when an invalid recipient is sent in <RCPT TO>, which keeps Exchange in command state unless valid recipient is sent. Hence, IDS can't decide the validity of the recipient and moves to DATA state whereas Exchange remains in command state. This is another successful IDS evasion.

Another example is on the BDAT flaw. BDAT accepts <SIZE> argument in decimal number. For a large value of <SIZE>, Exchange spins it between negative and positive value. For the Exchange server, the effective value of 4294967296 is 0 and 4294967296 + 100 is 100. While IDS tries to decode value 4294967297 which is just a value of 1, the malicious command is sent to the Exchange. The fact that the IDS doesn't behave and doesn't own the built-in function exactly as the Exchange server makes it possible to bypass the IDS as it focuses its resources to decode the number.

IDS Evasion Mitigation

Large numbers of published and exposed techniques have already been blocked and countered by IDS vendors. However, application layer

remains a large threat for attackers to exploit. As more applications are introduced and developed, the way to exploit and launch attack also increases rapidly.

Among the best solution to counter IDS evasion is to properly choose the type of IDS implemented and the features it supports. Proper placement of IDS which maximizes its effectiveness also helps a lot in detecting intrusion. According to VeriSign team (2006), the best approach to mitigate the threat of IDS evasion is to maintain security vulnerability awareness, patch vulnerabilities as soon as possible and wisely choose the IDS based on the network topology and network traffic received.

Normalization should be implemented properly and flexibly by the IDS. Normalization is a process that attempts to translate obfuscated input into the output that the end host will receive. By applying normalization, fragmented packets can be reassembled in proper order which allows the IDS to view the information as the target host sees it.

TTL problems can be solved by automatically changing the TTL field to a large value, which ensures that the end host always receives the packets (VeriSign, 2006). Another method that may solve the TTL problem is for the IDS to collect all the MAC addresses within the network it is monitoring. It must then obtain all the TTL information from each host and map the information. This allows the IDS to know the distance to the host for which the data is destined and also know the host's MAC address (VeriSign, 2006).

Detecting polymorphic shellcode may be a bit challenging. Nevertheless, one possible method is to look for 'nop opcode' other than '0x90'. The problem is that there are several 'opcodes' that do not access memory, but alter register values (VeriSign, 2006). This problem can be mitigated by triggering an alarm on certain threshold. However, it may also cause false alarms whenever the threshold condition is met in normal circumstances.

Administrators can also use security softwares such as Nmap and Nessus that are freely distributed which helps them to detect vulnerabilities within their entire network. The concept applied by these softwares is to attack ones own system in order to figure out the weak points and fix them up accordingly.

Conclusion

IDS is a useful security tool but not without weaknesses and limitations. In reality, there are no absolute and truly secure system; there are always ways to break in or manipulate the system. Furthermore, human errors, instead of system errors, often contributed a lot in security breach.

IDS alone cannot provide sufficient information to detect evasions. IDS should always be used in conjunction with other security tools such as firewalls, intruder prevention system (IPS) and anti-virus. Moreover, the selection of the type of IDS and its strategic placement within the system is required in order to maximize detection capability.

No matter how many ways there are to detect IDS evasion attempts, there are also equally as many traditional and new ways to elude IDS detection. The fundamental concepts of IDS evasion are still the same; insertion, evasion, and ambiguities attacks. These attacks can trick and bypass the IDS, be it host or network based, even up to this very day. Application layer attacks are currently an open playfield for unethical hackers. As new applications and softwares are introduced, new opportunities are presented for attackers to exploit the system.

The most viable approach in security in any environment is to always limit the avenues of attack. Any unnecessary service and path that are unnecessarily exposed should be closed off. Essential services should always be monitored closely with proper scrutiny. Administrators are also required to apply the latest software fixes and patches as quickly as possible. Sound knowledge on network establishment and technicality is a must if one is really serious on thwarting any kinds of intrusion. It is awareness that may eventually help mitigate IDS evasion activities.

Bace and Mell (2004) noted that the IDS product sales projected to reach USD 978 million by 2003 and growing. This shows strong commercial demand and a good future indicator for IDS. IDS still currently plays an equally important role in enforcing security within an organization and proves to be a very valid addition to the security block.

References

Axelsson, S. (2000). *Intrusion Detection Systems: A Survey and Taxonomy*. Retrieved 21 January 2009, from www.mnlab.cs.depaul.edu/seminar/spr2003/IDSSurvey.pdf.

Bace, R. & Mell, P. (2004). *NIST Special Publication on Intrusion Detection Systems*. Retrieved 18 January 2009, from http://csrc.nist.gov/publications/ nistpubs/800-31/sp800-31.pdf.

Basicevic, F., Popovic, M. & Kovacevic, V. (2005). The Use of Distributed Network-Based IDS Systems in Detection of Evasion Attacks. *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop*, 78-82.

Caswell, B. & Moore, H.D. (2006). *Thermoptic Camouflage: Total IDS Evasion*. Retrieved 19 January 2009, from www.blackhat.com/presentations/bh-usa-06/BH-US-06-Caswell.pdf.

DCE/RPC (2005). Retrieved 1 February 2009, from http://en.wikipedia.org/wiki/DCE/RPC.

Hatti, A. *Legiment Techniques of IDS/IPS Evasion.* Retrieved 19 January 2009, from www.clubhack.com/2007/files/Ajit-Legiment_Techniques.pdf.

HTML (2001). Retrieved 1 February 2009, from http://en.wikipedia.org/wiki/HTML.

Ingeborn, A. (2001). *IDS Evasion Design Tricks for Buffer Overflow Exploits*. Retrieved 21 January 2009, from www.blackhat.com/presentations/bh-europe-01/ingeborn/bh-europe-01-ingeborn.ppt.

IWS - The Information War Site (2001). *Obtaining and Cracking UNIX Password File (part 1)*. Retrieved 21 January 2009, from http://www.iwar.org.uk/hackers/resources/digital% 20rebels/cupw.htm.

JavaScript (2001). Retrieved 1 February 2009, from http://en.wikipedia.org/wiki/JavaScript.

Mutz, D., Vigna, G. & Kemmerer, R. (2003). An Experience Developing an IDS Stimulator for the Black-Box Testing of Network Intrusion Detection Systems. *Proceedings of the 19th Annual Computer Security Applications Conference*, 374-383.

Pastrana, S., Orfila, A. & Ribagorda, A. (2011). A Functional Framework to Evade Network IDS. *Proceedings of the 44th Hawaii International Conference on Systems Sciences*, 1-10.

Ptacek, T. & Newsham, T. (1998). *Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection, Secure Networks, Inc.* Retrieved 18 January 2009, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.5765.

Raffy (2004). *Attack Chains*. Retrieved 21 January 2009, from http://www.raffy.ch/projects/ Raffael_Marty_GCIA /node22.html.

Shellcode (2003). Retrieved 1 February 2009, from http://en.wikipedia.org/wiki/Shellcode.

The VeriSign iDefense Intelligence Team (2006). *Intrusion Detection System (IDS) Evasion*. Retrieved 18 January 2009, from http://whitepapers.techrepublic.com.com/ abstract.aspx?docid=361886.

Timm, K. (2002). *IDS Evasion Techniques and Tactics*. Retrieved 18 January 2009, from http://www.securityfocus.com/ infocus/1577.

Transport Layer Security (2001). Retrieved 1 February 2009, from http://en.wikipedia.org/wiki /Transport_Layer_Security.

Universally Unique Identifier (2003). Retrieved 1 February 2009, from http://en.wikipedia.org/wiki/ Universally_Unique_Identifier.