SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of James Paul Black:

Techniques of Network Steganography

and Covert Channels

J. C. Interlando, Chair
Department of Mathematics and Statistics

Stefen Hui
Department of Mathematics and Statistics

William Root
Department of Computer Science

11/3/2013
Approval Date
DEDICATION

To Rob, with gratitude for your unwavering support throughout the years. I wouldn’t be who I am today without your guidance and influence.
Alice opened the door and found that it led into a small passage, not much larger than a rathole: she knelt down and looked along the passage into the loveliest garden you ever saw. How she longed to get out of that dark hall, and wander about among those beds of bright flowers and those cool fountains, but she could not even get her head through the doorway...

– Lewis Carroll, *Alice’s Adventures In Wonderland*
ABSTRACT OF THE THESIS

Techniques of Network Steganography and Covert Channels

by

James Paul Black

Master of Science in Applied Mathematics with a Concentration in Mathematical Theory of Communication Systems
San Diego State University, 2013

Covert channels encompass a group of methods designed to alleviate the insufficiency of encryption by applying principles of steganography to hide both the data and method of communication. Covert channels transmit information in violation of a systems security policy; data transfer is typically achieved by modifying network protocol packets in such a way that data is covertly injected into a standard packet and allowed to travel across the network. An outside listener can then collect the packets, strip the injected information from the obtained packets, and retrieve the desired data. Covert channels may be viewed as either a boon or burden depending on the instance of use.

This paper starts by describing techniques of covert channels in depth, including various types of timing channels, storage channels, and hybrid implementations. A demonstration of Perl scripting in conjunction with commonly available free software will emphasize the relative ease of implementing a covert channel. Methods to mitigate attacks and prevent future attacks are presented and discussed. While many works within the field of covert channel analysis focus on creating specific covert channels, few works broadly identify channel implementations and prevention techniques, and none provide explicit coding demonstrations to showcase the relative ease of implementation. This thesis seeks to expand on previous research centered on broad analysis of covert channel implementations and provide a suitable coding demonstration.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Contribution of This Work</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Overview of This Work</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>SCENARIOS</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Scenario 1: The Prisoner Problem</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Scenario 2: Restrictive Locales</td>
<td>3</td>
</tr>
<tr>
<td>2.3</td>
<td>Scenario 3: Insider Threat</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>TECHNIQUES OF NETWORK STEGANOGRAPHY</td>
<td>5</td>
</tr>
<tr>
<td>3.1</td>
<td>Internet Protocol Suite</td>
<td>5</td>
</tr>
<tr>
<td>3.2</td>
<td>A Note on Channel Capacity and Efficiency</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>TIMING CHANNELS</td>
<td>9</td>
</tr>
<tr>
<td>4.1</td>
<td>Packet Timing and Rate Manipulation Methods</td>
<td>9</td>
</tr>
<tr>
<td>4.2</td>
<td>Packet Sorting Method</td>
<td>11</td>
</tr>
<tr>
<td>4.3</td>
<td>Packet Retransmission Method</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>STORAGE CHANNELS</td>
<td>13</td>
</tr>
<tr>
<td>5.1</td>
<td>Packet Retransmission Method</td>
<td>13</td>
</tr>
<tr>
<td>5.2</td>
<td>TCP/IP Header Modification Methods</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>EXOTIC IMPLEMENTATIONS</td>
<td>17</td>
</tr>
<tr>
<td>6.1</td>
<td>Ad-Hoc Routing Protocols</td>
<td>17</td>
</tr>
<tr>
<td>6.2</td>
<td>Temperature Based Method</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>ICMP STORAGE CHANNEL DEMONSTRATION</td>
<td>19</td>
</tr>
<tr>
<td>7.1</td>
<td>Ping Utility</td>
<td>19</td>
</tr>
<tr>
<td>7.2</td>
<td>Method of Implementation</td>
<td>20</td>
</tr>
<tr>
<td>7.3</td>
<td>Interpretation of Results</td>
<td>21</td>
</tr>
</tbody>
</table>
8 DETECTION AND MITIGATION ................................................................. 24
  8.1 Detection Techniques ................................................................. 24
  8.2 Mitigation Techniques ............................................................... 25
9 FUTURE ENHANCEMENTS AND CONCLUSION ............................... 27
BIBLIOGRAPHY ........................................................................... 29
APPENDIX
  ICMP Echo Request Covert Channel Source Code ............................. 31
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Representation of Internet Protocol Suite data encapsulation.</td>
<td>7</td>
</tr>
<tr>
<td>5.1</td>
<td>Representation of acknowledgement frame decomposition.</td>
<td>13</td>
</tr>
<tr>
<td>5.2</td>
<td>Representation of an IP packet header decomposition.</td>
<td>14</td>
</tr>
<tr>
<td>5.3</td>
<td>Representation of a TCP packet header decomposition.</td>
<td>15</td>
</tr>
<tr>
<td>7.1</td>
<td>Representation of an ICMP packet decomposition.</td>
<td>19</td>
</tr>
<tr>
<td>7.2</td>
<td>Representation of an ICMP echo request packet decomposition.</td>
<td>20</td>
</tr>
<tr>
<td>7.3</td>
<td>Screenshot of complete ICMP packet crafting script.</td>
<td>22</td>
</tr>
<tr>
<td>7.4</td>
<td>Screenshot of decrypting option within the script.</td>
<td>22</td>
</tr>
<tr>
<td>7.5</td>
<td>Screenshot of tcpdump captured packet.</td>
<td>23</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

I would like to thank Dr. Interlando for his enthusiasm, guidance, and support throughout the course of this research and thesis production. I would also like to thank Drs. Hui and Root for their helpful suggestions and for serving on my thesis committee. For all members of my thesis committee, your insight both inside and outside the classroom has been a source of great inspiration. Thank you.
CHAPTER 1
INTRODUCTION

The term *secure communication* is oftentimes associated with the encryption algorithms necessary to protect data and lines of communication; DES, AES, RSA, and NTRU are a few examples of cryptographic systems currently used to prevent attack on private data. As computing resources become more efficient in terms of cost and power, brute force attacks increasingly threaten data security. In response, algorithm developers increase encryption key size; DES evolves into Triple DES and cryptosystem failure is temporarily prevented. However, using an encryption method can draw unwanted attention from attackers by implicitly placing a value on the data. Covert channels encompass a group of methods designed to alleviate the insufficiency of encryption by applying principles of steganography to hide both the data and method of communication.

Coined by Lampson in 1973, the term *covert channel* was initially used to describe the inadvertent data leakage between processes on a monolithic system. Lampson identified that a computing process could create an observable effect on computing resources or leave a trail in system memory, unintentionally leaking data; this was referred to by Lampson as the problem of confinement. Covert channels became the term used to identify the illegitimate communication channels that leaked information in violation of system or process restrictions. [13]

Covert channels have evolved alongside developments in both networking and the Internet. In modern communication processes, covert channels can be used as a form of digital steganography to obfuscate the data transmission. Although initially used to describe data leaks across processes within a single computer system, the term has broadened to include data transfers across computer networks. Modern implementations have used network protocols in violation of their intended process to transfer information between nodes and off of the host network.

It should be noted that steganography typically refers to concealing information in content while covert channels refers specifically to hiding information in network protocols. However, both network steganography and covert channels have been used interchangeably with similar intent. Zander *et al.* note that such interchanging of terms was likely a result from conflicting definitions of covert channels by both Lampson and the United States Department of Defense [24]. Petitcolas *et al.* note that the term *information hiding* can be generically used to reference both digital steganographic techniques and covert channel...
Regardless of terminology semantics, covert channels represent a form of digital communication masking that employs principles of steganography.

1.1 Contribution of This Work

This paper starts by describing techniques of covert channels in depth, including various types of timing channels, storage channels, and hybrid implementations. A demonstration of Perl scripting in conjunction with commonly available free software will emphasize the relative ease of implementing a covert channel. Methods to mitigate attacks and prevent future attacks are presented and discussed. While many works within the field of covert channel analysis focus on creating specific covert channels, few works broadly identify channel implementations and prevention techniques, and none provide explicit coding demonstrations to showcase the relative ease of implementation. This thesis seeks to expand on previous research centered on broad analysis of covert channel implementations and provide a suitable coding demonstration.

1.2 Overview of This Work

- In Chapter 2, we describe potential scenarios in which an individual or group may choose to use covert channels as a communication medium.
- In Chapter 3, we identify the primary classifications of covert channel implementations, describe the component layers of the Internet Protocol Suite, and briefly discuss the metrics used to calculate channel capacity and efficiency.
- In Chapter 4, we describe covert timing channels in depth, including an analysis of specific timing channel examples.
- In Chapter 5, we describe covert storage channels in depth, including an analysis of specific storage channel examples.
- In Chapter 6, we describe more exotic implementations of covert channels in an effort to demonstrate that the potential for a covert channel medium is limited only by a software developer’s creativity.
- In Chapter 7, we describe the role of both the ping utility and the Internet Control Message Protocol within the Internet Protocol Suite, demonstrate a covert storage channel that operates in conjunction with ping manipulation, and demonstrate a method for observing the data transference.
- In Chapter 8, we identify methods of detecting and mitigating covert storage and timing channels.
- In Chapter 9, we identify future covert channel enhancements and conclude the research.
CHAPTER 2

SCENARIOS

As derived from standard cryptography convention, let Alice and Bob be communication partners and let Eve and Oscar represent attack vectors. As our final character, let Wendy be a warden who monitors all messages [20]. In the case of covert channels, consider Wendy to be a warden who monitors all traffic across the network. Covert channels may be considered a boon or burden depending on the intent of the user; while multiple variations of scenarios can occur, consider some of the following possibilities:

2.1 SCENARIO 1: THE PRISONER PROBLEM

In conjunction with Wendy, Simmons introduced the concept of the prisoner problem, considered by researchers to be the standard covert channel model and adaptable to many information hiding scenarios. Consider Alice and Bob to be incarcerated criminal associates. Both Alice and Bob are seeking to escape but doing so requires a coordinated effort and both reside in different cells within the jail. Wendy the warden allows prisoners to communicate but monitors all messages; if Wendy suspects any level of suspicious communication between Alice and Bob she will further confine them and prevent any chance of escape. Alice and Bob must therefore communicate covertly. [20]

In the case of electronically transmitted messages, Alice and Bob could benefit from steganographically communicating across the network to coordinate an escape. In this scenario, covert channels could be used to bypass Wendy and secure freedom. A scenario modification could involve Alice and Bob attempting to communicate sensitive information in a way that evades the efforts of Oscar and Eve.

2.2 SCENARIO 2: RESTRICTIVE LOCALES

Many nations, companies, and organizations restrict access to specific Internet content. Content filtering usually occurs via keyword search at the application level of internet traffic. If Alice is located in an environment known to have Wendy the content filtering warden, then covert channels can be implemented to bypass the content filters. Alice can use storage channels to covertly tunnel Internet packet requests to her proxy server located outside of the restrictive network. The proxy server then processes the web requests and sends the results back to Alice via the same covert channel. Since covert channels generally occur over the underlying layers of internet protocols, Alice’s traffic stands a good chance of bypassing Wendy’s restrictions.
Pragmatically, this means that if Alice is located in North Korea and can successfully ping a proxy server outside of the county, then she can gain worldwide Internet access despite government information restrictions. While covert channels may be a burden to network administrators, they can also be a boon to media and human rights agencies.

2.3 Scenario 3: Insider Threat

Bob is a disgruntled employee who decides to sell corporate secrets to the highest bidder. Bob suspects that Wendy automatically searches outbound emails for key phrases in an attempt to prevent an insider attack. Knowing that he stands to gain more by covertly leaking information off the server rather than overtly emailing a few documents to a competitor, Bob decides to use a covert channel approach to data exfiltration.

With high profile data thefts making headlines every month, a la Bradley Manning and Edward Snowden, it is prudent for network administrators to consider unusual communication channels and not just standard data transfer methods. In a similar version of this scenario, Bob could remotely implement malware designed to exfiltrate data from corporate server; command and control with malicious programs is oftentimes accomplished through the use of covert channels [11]. Fisk et al. note that if only 1 bit of covert information is transferred per packet, a large Internet business could lose as much as 26 gigabytes of data per year [7]. With a properly emplaced covert channel, 1 bit of corrupted data per packet can easily go unnoticed or falsely attributed to transmission errors. Understanding covert channels from an attacker’s perspective is necessary to prevent their misuse.
CHAPTER 3
TECHNIQUES OF NETWORK STEGANOGRAPHY

In 1983, the United States Department of Defense published the Trusted Computer System Evaluation Criteria (TCSEC); colloquially referred to as The Orange Book because of its inclusion in the DoD Rainbow Series, TCSEC established the standards required for evaluating security controls built into computer systems. While Lampson focused on inadvertent data leakage in violation of system policy, TCSEC instead redefined covert channel to include the *exploitation* of processes to transfer data in violation of system policy. In addition to the verbiage designating covert channels as a potential attack vector, TCSEC further categorized covert channels by type: timing channels and storage channels. Timing channels were defined to be inter-process communication methods achieved through the modulation of system resources. Storage channels were defined to be methods that would allow a process to write data to a storage location for later access by a different process residing outside of the security policy; both the writing and accessing could be direct or indirect and still achieve covert channel data transfer. [4]

While much has changed in computing since 1983, the principles and classifications of covert channels remain valid to this day. In a computer network environment, the TCSEC definitions can be further refined. Rather than solely focusing on inter-process communication on a single machine, timing channels can be defined to be the modulation of network transmission protocols designed to cause an effect across the observed network. Similarly, the refined definition of storage channels can become the methods that allow a process to modify transmitted network protocols to covertly carry data. It is possible to combine both methods to create hybrid channels. In all instances, covert channels operate by transmitting data against the implemented security policies of either the network hardware or the connected nodes.

3.1 INTERNET PROTOCOL SUITE

To understand how covert channels function, it is important to understand the architecture of network protocols. In a network environment, network protocols are the fundamental building blocks of the data transmission structure; all data movement across a network requires use of specific transmission protocols designed to facilitate communication between nodes or network hardware.
The Internet Protocol Suite (IPS) encompasses the suite of communication protocols that serve as an underlying model for network communications; note that techniques of network steganography primarily focus on manipulation of the IPS. IPS, commonly referred to as TCP/IP in deference to the two primary protocols contained in IPS, uses a quadruple layered protocol stack design to delineate between component levels of transmitted data. Later models of internet protocol architecture, such as the Open Systems Interconnection (OSI) Model, further delineate component layers within the standard four TCP/IP layers [18]. For simplicity, we will only consider the four layers of encapsulation found in IPS: the link layer, internet layer, transport layer, and application layer. [5, 6]

1. Link Layer: This is the most basic layer in the IPS protocol stack design and contains the protocols necessary to define the topology of the local network. Examples of link layer protocols used to define the type of local network transmission include Ethernet and the IEEE 802 Local Area Network standard.

2. Internet Layer: This is the second layer in the IPS protocol stack design and contains the protocols necessary to effect internetwork data communications. Examples of internet layer protocols used to define network to network communication include the addressing mechanisms IPv4 and IPv6.

3. Transport Layer: This is the third layer in the IPS protocol stack design and contains the protocols necessary to open and maintain connections to the appropriate application or process on the target host. An example of a protocol contained in the Transport Layer is the Transmission Control Protocol (TCP), which is used to order, error correct and transmit data across host to host connections.

4. Application Layer: This is the fourth layer in the IPS protocol stack design and contains the protocols necessary for process-to-process communications across networked devices. Examples of Application Layer protocols include Hypertext Transfer Protocol (HTTP) for web traffic, File Transfer Protocol (FTP) for host to host file transfers, and the cryptographic protocols Secure Shell (SSH) and Secure Sockets Layer (SSL).

Once all data has been appropriately encapsulated into the IPS protocol stack design, as depicted in Figure 3.1, a packet is formed. A packet is simply a formatted data unit that is capable of movement across the network. Note that a complete packet does not require all four layers; there are many types of packets that do not require the use of the application layer. Similarly, techniques of network steganography rely on various forms of packet manipulation, but may or may not require all layers of the protocol stack design.

Additionally, some of the following timing and storage channel techniques make use of IPv4 specifically. IPv4 is the fourth generation of the Internet Protocol addressing system; IPv4 uses a 32-bit address commonly identified by the dotted decimal address format, such as 192.168.1.1. Each of the four component blocks contains numbers normally selected between
1 and 255, which translates to 8 bits, or 1 byte, per component number. As IPv4 only contains a 32-bit address, the total number of unique addresses is limited to $2^{32}$; computationally speaking, the number of possible IPv4 addresses has been outpaced by the advent of personal network devices that each requires a unique IP address. While techniques have been employed to temporarily mitigate the IPv4 address shortage, true resolution is found in the IPv6 addressing scheme. IPv6 is the sixth generation of the Internet Protocol currently in early stages of unified adoption. Given the magnitude of network connected devices worldwide, IPv6 seeks to solve the shortage of unique network addresses by issuing a hexadecimal 128-bit address; this evolution of addressing represents a space of $2^{128}$ unique addresses. The resulting format change will require adapted network steganographic techniques for those covert channels relying specifically on IPv4 architecture. Unless expressly stated, all references to IP addresses in this thesis will be in the context of IPv4.

### 3.2 A Note on Channel Capacity and Efficiency

Channel capacity, or bandwidth, varies depending on the type of channel implemented. In storage channels, bandwidth is determined by the number of bit or byte modifications made within the manipulated network protocol. In timing channels, bandwidth is constrained by the volume of data traversing the network that is affected by the scheme of manipulation. Specific channel implementations are discussed within Chapters 4, 5, and 6.

When initially introduced by TCSEC, the Department of Defense stated that from a security perspective, covert channels with low bandwidths represent a lower threat than those with high bandwidths [4]. When determining data transfer rates, faster speeds or higher bandwidths may seem desirable. However, many intrusion or extrusion detection systems deployed across a target network may have rule based detection algorithms designed to
prevent suspicious data leaks. If Alice decides to use a covert channel with unusually fast transfer speeds or higher bandwidth, she risks detection or unwanted attention from Wendy or Oscar. In the case of overall efficiency, it would therefore be considered more efficient to transfer data at slower and less noticeable speeds; while individual communication transfer may be slower, the communicating parties stand to gain longevity of use of a particular channel and thus increase overall efficiency. Regardless of channel intent, software developers should be aware that fast connections can be detrimental to mission success; choosing to throttle an otherwise fast connection may make a difference when attempting to avoid detection.
CHAPTER 4  
TIMING CHANNELS

Timing channel communication methods rely on the modulation of system resources. For Alice and Bob to communicate through timing channels, they must first agree on a communication scheme; in a network environment, this can be accomplished by a variety of methods including manipulation of packet transmission timing, packet volume, or packet ordering. By construction, timing channels convey information through packet transmission patterns; it is therefore necessary for the sender and receiver to agree on timing parameters, and oftentimes achieve synchronization, in order to effect communication. Due to inherently lower capacity than storage channels, and the corresponding increase in detection difficulty, timing channels are an effective class of techniques. [2, 15]

Timing channels may be further categorized as active or passive depending on the type of impact they produce on the flow of digital traffic within the network. Active timing channels require the injection of packets to covertly transmit information, while passive timing channels modulate existing packet traffic. Active packet injection is inherently faster but greater transmission bandwidth is not always rewarded; difficulty in detecting passive packet modulation makes passive timing channels a potentially more robust option. [8]

While implementing these types of channels may seem straightforward, implementations of timing channels are complex and can experience degraded performance due to inadvertent transmission delays or deliberate prevention techniques. Cabuk et al. note that the efficacy of timing channels is dependent on network conditions, processing capabilities across the channel, algorithm complexity, and coding portability. Network congestion during peak hours of use may cause undesirable packet reordering, packet loss, or variations in round trip time between packet transmissions and responses. Similarly, both the sender and receiver must reduce extraneous impact on system resources to prevent processing delays. Finally, the sender and receiver should select efficient communication algorithms and ensure that coding functionality isn’t affected by differences within system environments. Identifying these constraints serves a dual purpose: software developers can now create more robust code, but network administrators can also implement targeted techniques to increase timing variations across their networks. [2]

While not an exhaustive list, the following are examples of timing channel implementations:
4.1 Packet Timing and Rate Manipulation Methods

One form of packet modulation transmits information through the manipulation of packet timing or packet rate. Both types of packet manipulation are protocol immaterial, meaning that the type of host protocol is a parameter selection left to the discretion of the end user(s). This allows a user to conduct reconnaissance of the target network, determine the most used protocols within that network, and select a similar stego-protocol in order to reduce detection.

Packet timing covert channels rely on modulating the delivery times of individual packets to receiving nodes in coordination with a preselected timing interval. In this instance, the timing internal represents the overall channel bandwidth; a narrower timing interval corresponds to a wider transmission bandwidth by allowing more data transfer within a fixed period of time. Once the communication nodes are synchronized, the sending system encodes a message into binary digits and transmits a corresponding stream of carefully timed packets. The receiving system then decodes the message by converting the presence of successive packets into a corresponding stream of binary digits. Demodulation is dependent on preselected and synchronized timing intervals; the presence of a packet within a timing internal may correspond to the bit 1 while no packet may correspond to the bit 0. Network packet transmissions are measured in milliseconds; in a low noise network environment, an active covert packet timing channel with a small timing interval may attain data transfer rates that rival the rates of overt communication methods.

Similarly to packet timing, packet rate covert channels rely on modulating the volume of packets delivered within a preselected and synchronized timing interval. An increase in packet volume may correspond to a 1 while a lull in packet traffic may correspond to a 0. Active packet rate covert channels operate by injecting large volumes of packet traffic, while passive packet rate channels manipulate the flow of traffic by throttling the bandwidth of the host channel.

Cabuk et al. are credited with implementing the first Internet Protocol (IP) timing channel. Some notable research contributions include bit modification, synchronization, and the previously discussed performance factors. Bit modification, achieved by appending additional bits to the transmitted bit stream, can be used for parity check redundancy, synchronization methods, and data encryption. Synchronization was discovered to be a more challenging issue. [2]

In their timing channel implementation, Cabuk et al. divide the message into smaller blocks of bits called frames. Synchronization is partially achieved through the use of start of frame (SOF) packets; the SOF packet alerts the receiver to monitor for incoming data during
the preselected timing interval. Once the data is received, and the frame transmission is complete, the receiver then enters a standby mode until the next SOF packet is detected. The use of silent intervals and gradual timing interval adjustments allows communication to continue despite changes in network use or congestion, however, network delays still cause some observable error in synchronization. With no error correction and standard 8 bit ASCII encoding, their covert channel implementation attained 98% accuracy with a timing interval of 0.06 seconds; this is equivalent to covertly transmitting 16.666 bits per second or approximately 1.852 characters per second given the inclusion of SOF bits. [2]

In response to the difficulty of synchronization, Zander et al. note that there are implementations of covert timing channels that do not require such efforts; one such timing channel method created by Sellke et al. uses the delay within successive packets to transmit covert information. In this case, the sender and receiver need only agree on delay intervals and the corresponding value. This type of channel is not limited to binary communication; the use of multiple delay intervals and corresponding values can allow for a multi-rate covert timing channel. [19, 24]

4.2 Packet Sorting Method

Kundur et al. note the possibility of varying the sequence number of transmitted packets to covertly convey information. It is important to note that the packet sorting method does not reorder the actual transmission of the packets, but simply alters the component sequence number field contained in IP Security protocols such as the Encapsulation Security Payload (ESP) or Authentication Header (AH). By modifying the component sequence numbers, Kundur et al. identify the availability of a covert channel that would likely go undetected by standard packet inspection software because the content of the packets remain unmodified. If there are n packets, then there are n! permutations of packet ordering; Kundur et al. estimate the maximum capacity of the packet sorting method to be \( \log_2(n!) \), that is, for n = 25 packets, approximately 83.7 bits can be covertly transmitted. Channel bandwidth is therefore determined by the number of packets used per transmission. [12]

Since both the sender and receiver must agree on a sorting scheme prior to transmitting, it is possible to enhance this method by defining a multi-rate encoding structure or augmenting the binary structure with more efficient error correction coding methods. Packet sorting retains an advantage over typical packet timing and rate manipulation methods because it does not require synchronization between the sender and receiver; the receiver’s concern is limited to the ability to intercept or log received a packet stream, which is considered to be a relatively simple task.

Other modifications to this scheme involve reordering the destination IP addresses of packets so that increasing or decreasing IP addresses can be used to covertly represent bits.
Another option involves modifying the sequence field to skip particular numbers, in effect an artificial packet loss; the sender and receiver need only define a mapping of lost numbers to corresponding bits. [24]

4.3 Packet Retransmission Method

In some protocol transmission schemes, a mechanism exists that forces the receiver to acknowledge the receipt of a packet; if a packet is not received within a predetermined time limit, an acknowledgement (ACK) is not sent, and the packet is automatically retransmitted. Note that valid ACK frames are meant solely as a control mechanism sent between communication partners [1]. The sender and receiver must agree on an encoding scheme; in a strictly timing based channel implementation, acknowledgements or failure induced retransmissions can correspond to timing intervals similar to packet timing methods. This type of packet retransmission based channel can also be modified to be either strictly storage based or a hybrid channel.
CHAPTER 5
STORAGE CHANNELS

Storage channel communication methods allow a process to modify transmitted network protocols to covertly carry data. For Alice and Bob to communicate through storage channels, they must again agree on a communication scheme; in a network environment, this can be accomplished by a variety of methods including direct modification of acknowledgement frames, header extensions and padding, or unused data fields contained within a protocol. By construction, storage channels effect communication by carrying the message itself within the confines of the modified protocol; due to inherently higher capacity than timing channels, and the corresponding increase in detection ability, storage channels are a less effective class of techniques. However, the relative ease with which storage channels can be implemented makes them a formidable method of covert information hiding, especially in a poorly monitored network environment.

Since storage channels rely on direct packet modification and injection designed to carry a covert message, they are considered to have an active impact produced on the flow of digital traffic within the network. While not an exhaustive list, the following are examples of storage channel implementations:

5.1 Packet Retransmission Method

Butti et al. note that packet retransmission acknowledgement frames can be used as a strictly storage channel based implementation; Figure 5.1 depicts a representation of the ACK
frame prior to manipulation. Butti et al. created a tool to encode messages directly into the receiver address component of the ACK frame, thereby covertly transmitting messages across the network. Several frames are needed to transmit a complete message, and the likelihood of detection is limited because it requires network administrators to inspect each packet and apply anomaly detection algorithms; in wireless network, ACK frames are so commonplace that they are likely to be ignored. While their technique focused solely on IEEE 802.11 wireless protocols, adaptations can be made for other similar implementations. [1]

Similarly to the packet retransmission methods previously discussed, one such hybrid method created by Mazurczyk et al. uses acknowledgement failures to invoke a retransmission request. During the course of normal communication, the sender and receiver exchange innocuous packets. However, during the connection, the receiver intentionally fails to transmit an ACK frame, triggering a retransmission request. The sender then replaces the original packet with a stego-packet that contains hidden information. The receiver then extracts the relevant information from the retransmitted packet. This method demonstrates a technique beneficial to successful covert channel implementation; by hiding information within a stream of innocuous data, Alice and Bob can evade Oscar’s detection through the assumption that Oscar will most likely inspect their overtly communicated data. [15]

5.2 TCP/IP HEADER MODIFICATION METHODS

TCP and IP headers that precede encapsulated data contain many component fields of data necessary to effect transmission across a network; for a decomposition of the TCP/IP header fields, refer to Figure 5.2 and Figure 5.3.

![Figure 5.2. Representation of an IP packet header decomposition.](image)

While most proposed TCP/IP header channels are relatively easy to implement, the overall bandwidth can vary wildly depending on the type of field selected and the type of bit manipulation employed. It may be possible to combine the below attack vector examples to create hybridization in an effort to reduce detection and increase bandwidth. Many authors
explore packet header field modification approach to storage based covert channels, including direct modifications of the following fields:

1. IP Type of Service (ToS): The IP ToS is an 8 bit field located within the IP header; it contains data necessary to determine the quality of requested service, including reliability, precedence, delay, and throughput. Manipulating the ToS field as a storage channel could allow a maximum bandwidth of 1 byte per packet, but doing so is ill advised because all 8 bits are never used simultaneously in a normal session. Hintz suggests the possibility of throttling the bandwidth by only modifying the delay bit to achieve a stealthier channel. [9]

2. TCP Initial Sequence Number (ISN): The ISN is a 32 bit sequential number that facilitates a network devices ability to index received data. Usually sequence numbers are random to prevent third parties from session hijacking normal communication streams. Manipulating the ISN field as a storage channel could allow a maximum bandwidth of 4 bytes, and since detection is less likely due to perceived number randomness, the entire field can be used as a covert channel. [10, 23]

3. TCP Urgent Pointer (URG): The URG field is a 16 bit sequence number that identifies the end of urgent data. The URG pointer is only interpreted if a specific URG control bit is set, so it is possible to avoid setting the control bit and use the remaining 15 bits for information hiding. Hintz notes that implementing such a channel is more difficult because the user should restrict the values in the URG pointer to maintain an appearance of actually pointing at a specific piece of data; furthermore, the URG pointer is never used without the URG control bit set, so the user risks relatively easy detection. [10]

4. Timestamp Fields: Timestamp fields are an optional data component in the TCP header and can be used to assist in determining transmitted packet order. Since most timestamps are not set to a system clock and are instead incremented by millisecond starting from a random value, the low bit in the timestamp appears to be random. It is possible to manipulate the low bit in the timestamp to instead be a stego-bit, with challenging detection due to the pseudo-randomness of the bit. Note that timestamps should be monotonically increasing to avoid unwanted attention or detection. It is also possible to implement a timing channel in which packet transmissions are modulated to generate the desired least significant bit; while no active modification occurs, the passive slowing effect may be detectable. [9, 10, 23]
Similarly, information hiding modifications have been made to the remaining fields not specifically discussed, including header extensions, destination port number, header length, time to live field, or by reordering header fields [22, 23]. The method of header implementation is limited only by the developer’s creativity.
CHAPTER 6
EXOTIC IMPLEMENTATIONS

In an effort to avoid detection, increasingly exotic implementations of covert channels have been developed. To demonstrate that the method of covert channel implementation is limited only by the developer’s creativity, consider the following examples:

6.1 Ad-Hoc Routing Protocols
An ad hoc wireless network operates outside of traditional managed network infrastructure by creating a cluster of nodes that facilitates communication by relaying data between connected nodes; oftentimes this is accomplished via a mode of the same name contained within the IEEE 802.11 standard. Since nodes can move dynamically within the confines of the wireless network, the Ad-hoc On-demand Distance Vector (AODV) protocol is used to create the routing tables identifying ideal paths between nodes. When a node transmits a connection request, all nodes within the ad hoc network respond; the AODV protocol determines the route between the sending node and the receiving node with the least number of hops between third party nodes. Li et al. propose four methods of implementing covert channels within the AODV protocol, including modifications to timing the route requests, the source sequence number in the route request, the lifetime field in the route reply, and the destination identification field in the route request. Essentially, changing the delays between successive route requests, modifying sequence numbers, or hiding information in the destination identification field are all similar to methods previously discussed and achieve varying degrees of success. For example, in a network of $n$ nodes, modifying the destination identification field yields a possible $\log_2(n - 1)$ bits of hidden information. [14]

6.2 Temperature Based Method
Murdoch notes that a temperature based covert channel can be created by inducing and measuring changes based on the Central Processing Unit (CPU) temperature of an intermediate host node. While Murdoch applies the concept to detecting nodes hidden in networks such as Tor, Zander et al. expand Murdoch’s original concept to include general purpose information hiding. To transfer hidden information, the sender converts a message to binary and sends a corresponding stream of requests to the intermediate node; by varying the volume of data, similar to the packet volume modulation method previously discussed, the sender can induce temperature changes in the intermediate host’s CPU. The CPU temperature increases when it receives a higher volume of requests, altering the skew of its internal clock;
the receiver then measures the response time of requests sent to the intermediate host to deduce the code initially sent. [16, 25]

In determining the capacity of the temperature based method, Zander et al. note that the noise effect of ambient temperature and humidity on the observed CPU temperature can be factored out due to the longer and more gradual nature of their impact. They note that other sources of observed noise tend towards a Gaussian distribution, and thus base their capacity estimates on the Additive White Gaussian Noise (AWGN) channel model. Their modified capacity estimate $C$ is:

$$C = B \times \log_2(1 + \frac{P}{N}),$$

(6.1)

where $B$ is the channel bandwidth, $\frac{P}{N}$ is the signal-to-noise ratio. In simulations with modern hardware, the realized low noise channel capacity is 20.5 bits per hour and the high noise capacity is 10.3 bits per hour. While this is not a high bandwidth covert channel, it is important to note the success and creativity of implementation. [25]
CHAPTER 7
ICMP STORAGE CHANNEL DEMONSTRATION

This thesis contains a demonstration of coding designed to showcase the simplicity of implementing a covert storage channel and the benefits of modifying the message with data encryption and obfuscation methods. This demonstration is based on manipulation of the Internet Control Message Protocol (ICMP). A core control protocol contained in the IPS internet layer, ICMP is used to send messages between network hardware relating to the functionality of the network itself. ICMP packets typically convey information regarding network strength or communication errors; Figure 7.1 depicts a decomposition of an ICMP packet.

![ICMP Packet Decomposition](image)

**Figure 7.1. Representation of an ICMP packet decomposition.**

7.1 PING UTILITY

Few user based applications rely on ICMP, but one well known program is the ping utility; ping is a network utility that tests the ability to reach a desired host over a network and measure the roundtrip time of communication packets between a node and the target host. Ping functions by sending and receiving specifically crafted ICMP packets called echo request packets as depicted in Figure 7.2. Ping has been historically abused in both Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks in which a large volume of ICMP echo request packets are sent as a method of overwhelming a target network or host; however,
the utility is oftentimes prepackaged within standard operating systems and is therefore a
readily available tool that can be used to craft custom packets.

<table>
<thead>
<tr>
<th>ICMP Packet Decomposition (IPv4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo Request or Reply</td>
</tr>
<tr>
<td>bit offset</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>16:23</td>
</tr>
<tr>
<td>24:31</td>
</tr>
<tr>
<td>Message Type Code</td>
</tr>
<tr>
<td>Identifier Sequence</td>
</tr>
<tr>
<td>Data</td>
</tr>
</tbody>
</table>

**Figure 7.2. Representation of an ICMP echo request packet decomposition.**

In this instance, the operating system used, Backtrack 5, contains the free ping utility
hping3 that was released under the GNU General Public License. Similar to most common
ping utilities, hping3 is a packet crafting and network analysis program designed to ping hosts
and sniff firewall settings. Hping3 has an additional benefit of being scriptable in conjunction
with user defined code; in this case, scripting was used to prototype a covert channel exploit
by automating a custom packet crafting implementation with included encryption and
obfuscation functions. Note that not all computing environments contain hping3, but common
operating systems should contain a similar utility; interested readers may choose to adapt the
attached code for other versions of operating systems or ping utility programs.

### 7.2 Method of Implementation

This method of ICMP echo request packet manipulation makes use of the optional
data field. Under the assumption that network administrators may automatically detect, filter,
and/or restrict anomalous packets of length greater than 1024 bytes, it is recommended to
reduce the overall size of the message to under 120 ASCII characters, although further
restrictions may yield better results. Additionally, this implementation makes use of a choice
of rotation (ROT) ciphers; a type of Caesar cipher, the ROT 13 cipher rotates input letters by
13 places, while the ROT 47 cipher rotates all ASCII case sensitive text and punctuation
marks. Two successive shifts of either ROT 13 or ROT 47 will return the original plaintext.
Other cryptosystems can be used in place of the ROT ciphers, such as DES, but generally
require external packages to implement; in this case, the chosen ROT cipher variations are
inline functions fully contained within the script. Finally, further obfuscation is applied by
converting the encrypted text to *Base64* in an attempt to mitigate perceivable plaintext
sentence structure. Since Base64 encoding returns padded values, the user is again reminded to avoid sending lengthy messages.

All coding is automated through the use of Perl scripting; similar implementations could be made with other scripting or programming languages as well. Scripting is advantageous because it decreases the time spent manipulating both the message and crafting the custom packet. Transmission is sent from the host node to a predetermined IP address located at the beginning of the code. The address in this case was randomly determined to be 192.168.2.148; the sender should elect to change this IP address in favor of the receiver’s specific IP address to allow for delivery. Again to reduce anomaly detection, the script specifically directs all traffic to port 80 on the receiver’s node because port 80 is used for communicating standard web traffic; however, the user may elect to send custom packets to any port, and can do so by altering the script. The receiver need only log all packet requests, a process made easier by filtering incoming packets based on the sender’s IP address.

In both sending and receiving, the user may choose to reduce verbosity by commenting out or eliminating the print commands associated with displaying encryption and Base64 manipulation; they are left active in the example to demonstrate the steps executing within the script.

7.3 INTERPRETATION OF RESULTS

To ensure that packets are transmitted, the packet sniffing program tcpdump will be used to collect all transmitted packets within the confines of my network. If the user prefers a graphical interface, packet sniffing programs such as Wireshark can be used with similar intent. In the Backtrack 5 terminal, tcpdump is initialized with the command “tcpdump -vv -X dest 192.168.2.148”. This means that the tcpdump output is defined to be very verbose and filtered to show only packets with the preselected 192.168.2.148 destination IP address. While not necessary, filtering greatly reduces the amount of captured packets to inspect; in a wireless environment, this reduction could mean the difference between visually inspecting thousands of packets or just the single packet of interest.

As shown in Figure 7.3, we begin this iteration of data encryption and transmission by calling the ICMP script. The ICMP script asks the user to select between encoding, decoding, and exiting; we select encoding by entering “E”. The ICMP script then asks the user to select the type of rotation cipher; we select ROT 47 by entering “47” as shown. The final call in this sequence is to enter the message; we enter “This is a covert channel.” as the plaintext message. The script automatically executes the selected ROT 47 encryption option, obfuscates the data with Base64 encoding, and prints each value as a check on process completion as shown. The script then automatically packages the encrypted and obfuscated message into the optional data section of the ICMP echo request using hping3; the resulting transmission of the
entire packet to the preselected destination IP address and transmission results are shown at the bottom of Figure 7.3. Note that no response packets are received since the IP destination address was fictitiously created.

Figure 7.3. Screenshot of complete ICMP packet crafting script.

The decode option for recovering the example plaintext is shown in Figure 7.4.

Figure 7.4. Screenshot of decrypting option within the script.

Had the IP address not been fictitious, the receiver would simply collect all ICMP echo requests originating from the sender’s IP address, strip the text from the data section, and process it through the script decode option to obtain the hidden message.
We return to tcpdump and inspect the results of the sniffed packets as shown in Figure 7.5 to verify that the ciphertext message was properly injected into the ICMP echo request packet and transmitted.

![Figure 7.5. Screenshot of tcpdump captured packet.](image)

Note that tcpdump returns three columns of data; from left to right, the three columns are the bit offset, the hexadecimal representation of the transmitted data, and the ASCII conversion of the hexadecimal data. In this example, the reader can inspect Figure 7.3 and Figure 7.5 and note that the ASCII converted text from the data section of the ICMP echo request is the same as the encrypted and obfuscated message. Curious network administrators may choose to use Base64 decode to inspect packets for anomalous data, however, doing so on this data will only return the encrypted text.
CHAPTER 8  
DETECTION AND MITIGATION

In principle, detection of covert channels should be relatively straightforward, but as Zander et al. note, experience shows that few security products exist to effectively manage covert channels [23]. This chapter will identify methods of detecting previously discussed covert channel techniques and will offer suggestions on mitigating covert channels.

8.1 Detection Techniques

Techniques to detect covert channels depend on the method of channel implementation; since communications methods vary wildly, detection efforts must be targeted to identify potential leaks. Detection involves identifying if a covert channel exists, and identifying both the sender and receiver [14]. While not an exhaustive list, consider some of the following effective detection techniques:

1. Timing Channel Detection: Network based anomaly detection can be successful against covert channels because it functions by finding patterns or trends within the data traversing the network. For example, Gianvecchio et al. propose a detection system that measures changes in entropy within the network to determine if timing channels are being used, although more recent channel implementations have been designed to mimic standard entropy rates to remain undetected [8, 19]. Anomaly detection can be also used to find traffic latency in packet timing or rate; Cabuk et al. note that inter-packet delays are approximately normal, so statistical variance methods can be used to determine anomalous patterns created by those channel schemes [2]. This technique may also be useful against temperature based methods, if detection is used to determine if anomalous packet rates coincide with CPU timing clock skew and probes of the host clock. Implementations of anomaly detection against timing channels are less challenging since the program needs only identify patterns within the flow of network data, not within individual packets themselves.

2. Storage Channel Detection: Similar to detecting timing channels, anomaly detection can be used to detect implementations of storage channels. ACK frames could lead to detecting anomalous patterns created by Butti et al. In terms of header manipulation, most authors identify that anomaly detection could identify with moderate ease any modifications made to header padding, URG pointer, ToS field, and the timestamp field. Poor implementations by developers seeking greater bandwidth at the cost of detectability, like using the entire ToS field, make the task of detection substantially easier. However, for an anomaly detection system to function properly, it must be allowed to conduct deep packet inspection (DPI) in order to detect any potential patterns within all possible header fields. These systems generally result in degraded
network performance, so oftentimes intrusion detection systems (IDS) that use DPI are restricted to only search for attack vectors deemed to be most likely; this is a significant security flaw, and can be bypassed by a knowledgeable attacker who tailors their attack method after first conducting a penetration test of the network to see which modified packets get overlooked.

3. Statistical Deviance: ISN methods of covert data hiding are considered more secure, but still fall victim to poorly designed coding. Tumoian et al. propose a method of measuring statistical deviance after implementing a ISN generation algorithm; any deviance from the generated ISN is regarded as an attack. This method could be applied to other fields contained within headers, especially the timestamp field. [21]

8.2 Mitigation Techniques

Due to difficulties in detecting an implemented channel, especially if the channel falls within the noise floor of network traffic similar in appearance to the stego-protocol, efforts to mitigate channels are easier to implement. Integral to covert channel mitigation is the idea that covert channel vectors must be identified in order to be prevented; regardless of how a network administrator determines attack vectors, they can only attempt to prevent those vectors of which they are aware. Failure to properly identify channels will result in an incomplete solution. Once channels are identified, consider some of the following options:

1. Channel Documentation and Auditing: Documenting the channel is the first step in mitigating covert channel implementations within a network. By documenting covert channels, network administrators can then monitor those channels for signs of intrusion using previously mentioned anomaly detection techniques. Both documenting and auditing may serve to dissuade would-be attackers from using those methods, although a creative attacker will simply look for another avenue of approach, especially in conjunction with a more exotic and overlooked vector. [24]

2. Bandwidth Limiting: Another option to mitigate covert channels is to throttle the bandwidth of identified channels. Once capacity estimates have been formed based on the number of bit manipulations, efforts to reduce that capacity can be implemented. Packet jammers and modification programs can inject noise or specific values into the header storage fields previously discussed, in effect reducing the available bandwidth of the channel. Timing channels can be subverted by introducing timing mechanisms that inject additional noise into the channel. While covert channels may still exist within those confines, the perceived bandwidth is low enough to make them unenticing to would-be attackers. However, future covert channel enhancements with error correcting codes and channel hopping implementations may allow an attacker to effectively maximize use of lower bandwidth options. [24]

3. Channel Elimination: A final option of mitigation involves directly removing the availability of the channel. In the case of the ICMP demonstration, the access list code “access-list 101 deny icmp any any 8” can be added to a firewall to block all ICMP access at the router level. Since ICMP traffic from the Internet Service Provider (ISP) is
useful, an additional access rule “access-list 101 permit icmp xx.xx.xx.xx 0.0.0.255 any 8” can be added to the previous command to allow only traffic from xx.xx.xx.xx, the ISP’s IP address. Note that in a wireless environment, it may still be possible for Alice to send an ICMP echo request with an appended message to the ISP while Bob uses wireless packet capturing tools to retrieve the message; network administrators will need to approach covert channel elimination as creatively as an attacker would approach covert channel implementation. Other forms of channel elimination are more drastic and follow the DoD Sensitive Compartmented Information Facility (SCIF); in such an environment, all computer nodes access only one restricted network with no Internet access, all nodes are located within an equivalent vault, and access to those nodes is restricted to only those individuals permitted to do so. While this may not be effective at a normal business, similar efforts should be taken to restrict sensitive information to an intranet environment with no Internet facing front end. [6, 24]
CHAPTER 9
FUTURE ENHANCEMENTS AND
CONCLUSION

Due to increasing awareness of information hiding techniques, future implementations would benefit from some of the following techniques:

1. Encryption: Techniques of network steganography and cryptology do not need to be mutually exclusive; inclusion of encryption techniques could make the difference between safe transit of data and detection. As network administrators increasingly rely on algorithm based detection techniques, adding a second layer of protection through encryption would dramatically increase the security of the communication system since decoding the message would require a thorough understanding of both search algorithms and cryptanalysis. As shown in the ICMP channel demonstration, adding encryption or other obfuscation methods is as simple as adding a few additional lines of code to act on the message prior to transmitting.

2. Error Correcting Codes: Of particular relevance to covert timing channel methods, error correcting codes can be implemented to create more efficient transfer schemes; some timing channels already employ error correcting codes, but greater adoption could increase the effectiveness of covert channel methods. Many timing channels are considered one way communication channels because they only allow communication from the sender to the receiver. To ensure the message is transferred, especially when the sender is unable to confirm message receipt, error correcting codes should be implemented to reduce the impact of noise on the system. In covert channel implementations that rely on low bandwidth transmissions for stealth gains, especially since bit modification schemes, error correcting codes would be a wise addition.

3. Phase Locked Loop (PLL): A phase locked loop is a control system method containing a feedback loop designed to maintain communication channel synchronization. PLLs are widely used in many communication systems, especially in radio communications and digital modems. Timing channels that require synchronization between the sender and receiver may benefit from more advanced methods such as PLLs. Channels that employ these methods may be more challenging to detect; since the easiest timing channel implementations avoid the use of synchronization, especially inter-packet delay variants, returning to properly synchronized systems may enjoy a period of security through obscurity. [2]

4. Passive Timing Channels: Many timing channels discussed required active packet injection or modification to achieve communication. Passive systems that modulate existing traffic are harder to detect, especially when emplaced within the network rather than within the node. For example, if Alice emplaces a passive timing channel within
the network router, it would be substantially more difficult for a network administrator to detect since all traffic would be modified rather than traffic modifications stemming from a sole node. Gaining and maintaining root access to network hardware could be a viable method of covert channel emplacement.

5. IPv6 Compliant Channels: Full adoption of IPv6 will render most covert storage channel techniques and some timing channel implementations obsolete. Fortunately, the IPv6 header contains numerous component fields that can be similarly modified to carry hidden information; Qu et al. note that the time to live field in IPv4 will be replaced by a similar hop limit field in IPv6, allowing for portability of techniques between IP generations. Future work should involve adapting previously researched techniques to function within the next generation IP scheme. [22]

6. Port Knocking: Similar to the childhood secret door knock, port knocking is a network authentication technique in which a requesting node knocks on a particular sequence of closed ports to gain entry though an otherwise closed firewall. The host system actively monitors all port requests, and once the proper combination from a single node is received, the host allows the node to connect through the firewall using a predetermined protocol. Improvements to port knocking have been suggested by deGraaf et al., including using cryptography techniques and methods to overcome packet reordering [3]. Similar techniques could be employed to create a covert channel in which the sender port knocks an encoded message onto the receivers firewall; the receiver would only need to filter the observed knocks to isolate those originating from the sender.

7. Channel Hopping Methods: Greater research into channel hopping methods is needed. Similar to spread spectrum communication methods, channel hopping covert channels that spread data transmissions across protocols or time could serve to increase bandwidth while decreasing single channel detectability.

In conclusion, this thesis has explored techniques of information hiding, including timing and storage channels, and has demonstrated the relative ease of implementing such a channel. As long as computers are used, methods to covertly communicate data will continue to exist and thrive. In an age mired in reports of digital espionage and rampant cyber crime, covert channels may increasingly augment traditional overt methods of communication in an attempt to maintain protection from opponents. Regardless of whether a user seeks to implement new techniques, a network administrator seeks to further reinforce their digital defenses, or a developer seeks to create new exotic attack vectors, understanding network steganography is vital to achieving those goals.
BIBLIOGRAPHY


APPENDIX

ICMP ECHO REQUEST COVERT CHANNEL
SOURCE CODE
ICMP ECHO REQUEST COVERT CHANNEL
SOURCE CODE

########################################################################
# FILENAME ICMPcc.pl
# #
# DESCRIPTION ICMP echo packet covert channel with #
# simple ROT13, ROT47 encryption and #
# base64 obfuscation.
# #
# AUTHOR James P. Black
#
########################################################################
#!/usr/bin/perl
use warnings;
use strict;
use MIME::Base64;

# Set destination IPv4 address
my $ip = "192.168.2.148";

# ROT13 coding
sub rot13
{
    my $str = shift;
    $str =~ tr/A-Za-z/N-Za-m/;
    return $str;
}

# ROT47 coding
sub rot47
{
    my $str = shift;
    $str =~ tr/!/~/P~/O/;
    return $str;
}
print "Select Encode, Decode, or Quit. Enter [E/D/Q]: ";
chomp(my $input = <STDIN>);

#if, elsif, else statements
#Option to encode and transmit
#if input is "E" or "e"
if ($input =~ /[E]/i) {
    print "Encode option selected.\n";
    print "Select desired version of rotation cipher.",
        " Enter [13/47]: ";
    my $num = <>;

    if (($num != 13) && ($num != 47)) {
        print "Error: Invalid Entry. \n";
        exit;
    } elsif ($num == 13) {
        print "Enter text to encode and transmit: \n";
        my $text = <>; print "\n";
        my $rtext = rot13($text);
        print "ROT13 text: $rtext \n";
        my $b64text = encode_base64($rtext);
        print "Encoded text: $b64text\n";
        system("/usr/sbin/hping3 −c 1 −p 80 −n $ip −e $b64text");
        sleep 2;
    } elsif ($num == 47) {
        print "Enter text to encode and transmit: \n";
        my $text = <>; print "\n";
        my $rtext = rot47($text);
        print "ROT47 text: $rtext \n";
        my $b64text = encode_base64($rtext);
        print "Encoded text: $b64text\n";
        system("/usr/sbin/hping3 −c 1 −p 80 −n $ip −e $b64text");
        sleep 2;
    }
}
# Option to decode
# if input is "D" or "d"
elsif ($input =~ /[D]$/) { 
    print "Decode option selected.\n";
    print "Select type of rotation cipher used in transmission. Enter [13/47]: ";
    my $num = <>;

    if (($num != 13) && ($num != 47)) {
        print "Error: Invalid Entry. \n";
        exit;
    }

    elsif ($num == 13) {
        print "Enter received text to decode: \n";
        my $text = <>; print "\n";
        my $b64text = decode_base64($text);
        print "Rotation encoded text: $b64text\n";
        my $rtext = rot13($b64text);
        print "Recovered text: $rtext \n";
    }

    elsif ($num == 47) {
        print "Enter received text to decode: \n";
        my $text = <>; print "\n";
        my $b64text = decode_base64($text);
        print "Rotation encoded text: $b64text\n";
        my $rtext = rot47($b64text);
        print "Recovered text: $rtext \n";
    }
}

# Option to terminate script
# if input is "Q", "q", or blank
elsif ($input =~ /[Q]$/) {
print "Exiting.\n"; exit;

#else
print "Error: Invalid Entry.\n"; exit;

#endif