A Novel Covert Communication Method for Efficient Analysis of Network Vulnerabilities

by Jaime C. Acosta, Ph.D and John Medrano, Ph.D

ARL-TR-5926

February 2012

Approved for public release; distribution is unlimited.
NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.
A Novel Covert Communication Method for Efficient Analysis of Network Vulnerabilities

Jaime C. Acosta and John Medrano
Survivability/Lethality Analysis Directorate, ARL
**Abstract**

For an analyst to more efficiently conduct vulnerability assessments on networked systems, it is critical that their threat computer network defense tools work in a distributive fashion. The communication must be covert, unlike encryption, which only makes information unreadable to unintended recipients. In the past, covert communication in unstructured communication networks such as the Internet has been researched extensively; however, methods are still needed for covert communication in structured networks, such as wireless Army systems. In this report we describe the network blending communication system (NBCS) that blends messages with active network traffic in order to bypass an intrusion detection system. We evaluate the NBCS with an experiment and our results show that the method is reliable and that throughput and observable detectability can be parameterized.

**Subject Terms**

Covert communication, network blending, analysis

---

**Security Classification of:**

<table>
<thead>
<tr>
<th>a. Report</th>
<th>b. Abstract</th>
<th>c. This Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
<td>Unclassified</td>
<td>Unclassified</td>
</tr>
</tbody>
</table>

**Limitation of Abstract:**

Unclassified

**Number of Pages:**

34

**Name of Responsible Person:**

Jaime Acosta

**Telephone Number:**

(575) 678-8115
Contents

List of Figures v
List of Tables v
Acknowledgments vi

1. Introduction 1

2. Related Work 2

3. Preliminary Analysis 3

4. The NBCS 5
   4.1 Analysis Subsystem ................................................................. 6
      4.1.1 Analysis Configuration .................................................. 6
      4.1.2 Network Device Ifx ......................................................... 7
      4.1.3 Filter ........................................................................... 7
      4.1.4 Connection Manager and Timer .................................... 8
      4.1.5 Connection ................................................................. 8
      4.1.6 Rate Calculator ............................................................ 8
      4.1.7 Complexity Calculator ............................................... 8
   4.2 Communication Subsystem .................................................. 10
      4.2.1 Communications Configuration ................................... 10
      4.2.2 Covert Sender ............................................................ 11
      4.2.3 Covert Receiver .......................................................... 12
   4.3 Graphical Subsystem .......................................................... 13
      4.3.1 Statistics Display ......................................................... 14
      4.3.2 Connections Display ................................................... 15
      4.3.3 Byte Complexity Display ............................................ 16

5. Evaluation 16
   5.1 Experimental Setup ............................................................ 16

6. Results 18
List of Figures

Figure 1. Covert communications concept. ................................................................. 3
Figure 2. Dataflow between the subsystems of the NBCS. ........................................... 5
Figure 3. Dataflow between components in the NBCS. ............................................... 6
Figure 4. Complexity calculation. ................................................................................ 9
Figure 5. Sample histograms for different byte frequency distributions. .................. 10
Figure 6. Covert data insertion process....................................................................... 12
Figure 7. Screenshot of the connections display and byte complexity display. .......... 14
Figure 8. Statistics display GUI screenshot. ................................................................. 15
Figure 9. Throughput graphed against Byte Rate Thresholds. ................................. 19
Figure 10. Misses graphed against Byte Randomness Threshold. ............................. 20
Figure 11. Packets received graphed against Byte Rate Thresholds. ....................... 21

List of Tables

Table 1. Configuration parameters for the analysis components. ............................... 7
Table 2. Configuration parameters for the communications components. ................. 11
Table 3. Network traffic loads during experiments 1 and 3 (center column) and experiment 2 (rightmost column). ............................................................. 17
Acknowledgments

We would like to thanks Brenda Medina, Juan Ulloa, and the members of U.S. Army Research Laboratory (ARL)/Survivability/Lethality Analysis Directorate (SLAD) team for their helpful suggestions.
1. Introduction

For an analyst to more efficiently conduct vulnerability assessments on networked systems, it is critical that their threat computer network defense (CND) tools work in a distributive fashion. Example commercial-off-the-shelf (COTS) tools include vulnerability scanners (1, 2), network mappers (3) and protocol analyzers (4). These tools must coordinate through communication that is hidden so as to not alert intrusion detection systems (IDS). In other words, the communication must be covert, unlike encryption, which only makes information unreadable to unintended recipients. In the past, covert communication in unstructured communication networks, such as the Internet, has been researched extensively; however, methods are still needed for covert communication in structured networks, such as wireless Army systems.

Current covert communication methods, using storage and timing channels, do not exploit the properties of structured networks. Storage channels typically use properties of a protocol that are ignored, such as unused header fields. In this case, once the vulnerability of the protocol is documented, an attacker may uncover the data and breach the communication. Timing channels work by purposely modifying timing mechanisms on a network such as packet arrival times. In general, timing channels are difficult to detect, but they provide low throughput. In this paper we describe a novel blending method that is capable of using as carriers the payload fields of multiple connections including audio, video, and voice over Internet protocol (IP) (VoIP) streams.

To send covert data, our method executes in three main phases. In the analysis phase, the covert sender will analyze traffic in promiscuous mode. In the selection phase, the sender selects locations to place covert data. In order to blend with active network traffic, the sender will select connections with high data rates and a sufficient amount of randomness in the payload. Within these connections, the packets with the highest randomness (considered injection points) are duplicated and slightly modified to include the covert data. Finally, in the sending phase, the modified packets are injected into the network (still containing the original source and destination addresses). By analyzing the same traffic, the covert receiver will identify the injection points and extract the covert messages. We implemented the network blending communication system (NBCS) tool and evaluate it using user datagram protocol (UDP) connections during two network loads. Our results show that our method works with limited data loss and we analyze the trade-offs between throughput and observable detectability.

The rest of the report is organized as follows. First, related covert communication methods and tools are described. Second, the process that led to the development of the NBCS is provided. Third, the details of the system are described along with the data flow between components. Lastly, conclusions and plans for future work as reported.
2. Related Work

There are two main types of channels that are used for covert communication (5). The first are timing channels, which work by purposely modifying timing mechanisms on a network such as packet arrival times. In general, timing channels are difficult to detect, but they provide low throughput. The second type of channels are storage channels, which insert covert data inside header or footer fields in specific protocols, as described in (6, 7), and within payload fields of invalid messages. For example, the hidden communication system for corrupted networks (HICCUPS) (8) works by hiding data within the payload of messages with bad checksums in the datalink layer. This method works for wireless networks and requires specialized hardware that has the capability to modify data link layer checksums. In general, once these channels are documented, an adversary may know where to find these.

Recent methods use a combination of timing and storage channels. The Lost Audio PaCKets Steganography (LACK) method (9) works by having a legitimate VoIP stream between nodes on a network. The sender will then hide data in purposely delayed packets. In a VoIP stream, messages older than a certain period are usually ignored because each message generally carries only a few milliseconds of audio. It would not make sense for a receiver to process older messages, especially in streams with high message rates. Only a covert receiver captures and extracts the covert data within these messages. From a defensive perspective, just as a covert receiver monitors the old messages, an adversary could potentially do the same and, therefore, breach the covert communication.

One possible solution for this problem is to blend covert communication with normal traffic. In the field of system vulnerability analysis, one method that uses blending to bypass IDSs that identify anomalies in payload data is the polymorphic blending attack (PBA) (10). PBA works by monitoring traffic and identifying byte frequencies. In order to send malicious packets without triggering an anomaly detector, PBA pads the malicious data with bytes that make the entire contents match the known byte frequencies.

Applying this blending idea to covert communication requires additional considerations. First, covert communication requires a sender and a receiver, so the data must be coded in a way that a receiver can identify and extract the messages. In addition, assuming that invalid messages are not ignored, a covert communicator should modify only portions of messages that will not be noticed. Yarochkin et al. (11) demonstrate a method that creates connections that blend with network traffic (including protocols and services used). These connections are used to communicate covertly, and they are created and removed dynamically as the network changes. The method relies on having a network structure where new connection attempts are common and failed connection attempts are blocked and ignored, such as in Internet traffic. This is not the case in tactical networks where all nodes and connections among nodes follow some
specifications. Huang et al. (12) present another method that blends covert communication with existing traffic by hiding information in VoIP streams. In this work, the authors use the least significant bits (LSBs) in VoIP streams to hide data.

The problem with this is that the LSBs are used regardless of the data that are being sent. One problem that can occur is if the LSBs are unchanging, when inserting covert data the covert communication may become obvious. An extension of this work (13), improved the covert insertion process by selecting the LSB most similar to a given covert message to improve the blending (or transparency). The latter two methods use a single protocol and use fixed fields in the payload for covert communication. The method described in this paper chooses insertion points dynamically and embeds covert data over multiple connections.

3. Preliminary Analysis

Before it was implemented, the envisioned NBCS system would work by having two covert communicators listening promiscuously to network traffic. A covert sender would determine optimal locations to embed data. Using the same algorithm, a covert receiver would retrieve the data by looking in the same optimal locations (see figure 1). The implemented system required slight modifications as revealed with a preliminary analysis of common network traffic.

![Covert communications concept.](image)

The purpose of the analysis was to answer the following questions:

1. Is it possible for two or more covert entities to communicate by hiding data in optimal locations in network traffic?

2. What types of communications are better candidates for blending covert data?

A test network was set up that consisted of eight nodes (laptops in this case) that were connected to a hub with an average traffic load of 25 Mbps. Six nodes were overt communicators, meaning
that their identify was known, while two nodes were covert. In order to answer Q1, an informal test was conducted to determine whether two nodes could synchronize, i.e., whether nodes simultaneously captured the same traffic from the network. Wireshark was used to capture traffic on the two covert nodes, which had mirrored hardware and software.

After each second, the captured traffic was compared. Although most the same traffic was seen on both computers, with the exception of some missed UDP packets (less than 1%), it was not simultaneous, but within a 4-s window. For this reason, the implemented NBCS includes header and footer bytes with each covert packet. In order to answer Q2, the next step was to determine possible blending points by identifying connections in the network. A connection consists of the one-way packets between two nodes. These packets also use the same protocol and have the equal length. The following traffic protocols are among those captured during the analysis.

- Address resolution protocol (ARP)—for address resolution
- Internet control message protocol (ICMP)—a continuous ping between entities
- Domain name system (DNS)—domain name resolution
- Hypertext transfer protocol (HTTP)—web browsing
- Transmission control protocol (TCP) (raw)—remote desktop streams
- UDP (raw)—used for streaming audio using the Real-time Transport Protocol

To determine possible candidates for blending, at 1-s intervals complexity was measured for each connection by calculating the standard deviation and probability density of each byte (section 4.1.7 describes these methods in more detail). Also, the update rates (packets per second) for each protocol were recorded.

Keeping in mind that a potential adversary sniffing the network could attempt to breach the covert communication; candidate protocols for blending would have to exhibit high update rates and high amounts of complexity. Analysis of the traffic revealed that connections using TCP and UDP streams had the highest data rates and complexities. ARP and ICMP messages had the lowest complexities and rates. DNS had slightly higher complexities, but still had low rates. HTTP was sporadic, depending on whether streaming content or static web pages were being viewed through a web.

In summary, the following were learned during the analysis:

- Even in promiscuous mode, nodes on a network do not simultaneously capture the same traffic; therefore, header and footer fields must be appended to the covert data in order for a covert receiver to more reliably retrieve the data.
- Data stream traffic is most well-suited for embedding covert data due to high update rates and high complexity.
• Some protocols, such as TCP, use sequence numbers for control purposes. Embedding covert data in these types of connections will be seen as duplicates in wireshark. Many duplicates may alert an adversary; therefore, unreliable connections, such as UDP, are favored.

4. The NBCS

The NBCS (see figure 2) consists of three main subsystems along with a configuration component.

![Figure 2. Dataflow between the subsystems of the NBCS.](image)

The analysis subsystem captures overt and covert data on a network in promiscuous mode. Promiscuous traffic can be captured if the network nodes are connected on a hub, a switch with promiscuous ports, or a wireless device. The analysis subsystem analyzes selective traffic, such as TCP, UDP, etc., depending on the configuration, to determine covert data insertion points. Insertion points are bytes in selected packets that exhibit a high amount of complexity. The analysis subsystem also transmits packets with covert data onto the network.

The display subsystem is the graphical front-end component that displays various analysis functions to a covert communicator. Although this component is optional, i.e., the system may be run with only the command line interface to improve performance; this component is meant to help communicators that are unaware of properties of the traffic on the network, such as network load, communication protocols used, update rates, and number of connections. Communicators may then decide whether the current network is suitable for covert communication. Also, the graphical user interface (GUI) can be used to determine configuration values to use when choosing insertion points.

The communications subsystem inserts covert data into insertion points and also extracts covert data from incoming network packets. This component contains a configuration that gauges
which packets to search for covert data, allowing communicators to balance performance and data loss.

The following sections provide a walkthrough of the system detailing the dataflow between the components (as seen in figure 3).

![Dataflow diagram between components in the NBCS.](image)

Figure 3. Dataflow between components in the NBCS.

### 4.1 Analysis Subsystem

#### 4.1.1 Analysis Configuration

The analysis configuration component is a datastore that contains several parameters that may be adjusted to balance between system performance and capability. Table 1 contains a list of the parameters and a description of each.
Table 1: Configuration parameters for the analysis components.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Used By</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window Size</td>
<td>Connection Manager</td>
<td>Determines how often the packet counts, byte counts, and complexity values are computed. Also defines the smallest wait time before sending a covert message for a connection. For example, if this value is set to 1 s, for each connection, only one covert message can be sent with each second.</td>
<td>[100ms, 5000ms]</td>
</tr>
<tr>
<td>Interfaces to Use</td>
<td>Filter</td>
<td>Indicates which network device to use to capture data. Data from unspecified devices will not be processed.</td>
<td>[eth, wlan, any]</td>
</tr>
<tr>
<td>Protocol to Use</td>
<td>Filter</td>
<td>Indicates which network protocol(s) must be used by a connection to be a candidate for covert message insertion.</td>
<td>[UDP, any]</td>
</tr>
</tbody>
</table>

The configuration is implemented using xml files that are read at startup time. The values are stored in a singleton object.

4.1.2 Network Device Ifx

When a covert sender connects to a network, the DeviceIfx component will initialize the network interface devices and set them to operate in promiscuous mode. The jNetPcap application programming interface (API) is used for capturing traffic and also for injecting traffic (giving the ability to modify media access control (MAC) addresses, IP addresses, and other fields in the packets) on the network.

Regarding the implementation, handlers for the network devices are kept in a hash table and each handler listens for traffic on a separate thread. When a packet is received, it is passed to the filter component.

4.1.3 Filter

Depending on the configuration parameters, traffic from certain devices may be filtered or ignored. For example, covert communicators may wish to communicate only using certain wireless interfaces (as covert communication may be distributed across multiple subnets), so traffic seen on Ethernet interfaces may be ignored.

Specific protocols may also be filtered. A communicator may wish to only use UDP packets to decrease the load on the system. Also, whereas the UDP is connectionless and unreliable, TCP and other protocols use control mechanisms to ensure that all messages are received. Because the NBCS works by modifying and then replaying packets, using these messages for insertion points will result in messages with duplicate sequence numbers.

Depending on the network setup, this may or may not be acceptable. Regarding the implementation, when a device is filtered, the thread that handles the traffic is suspended. When a packet is received, the header is decoded using the jNetPcap API. If the protocol is not of a
desired type, it is ignored. Otherwise, the packet is passed to the connection manager component. There is one connection manager component for each network device.

4.1.4 Connection Manager and Timer

The connection manager contains connections. When a packet is received, the corresponding connection is updated. The connection manager also contains a timer that triggers at fixed time intervals (referred to as windows). The window size is defined in the analysis configuration. The number of bytes and packets seen during each window are also stored.

Regarding the implementation, the connections are stored in a hash table. When a decoded packet is received, the string concatenation of the header fields is used as the key. For each packet, if a connection is not already in the hash table, it is added, otherwise the packet is passed to the corresponding connection.

4.1.5 Connection

A connection consists of received packets that share the same source and destination MAC address, IP address, and port number. In addition, packets also share the same protocol (TCP, UDP, etc.) and length (number of bytes). Consequently, connections are unidirectional. Between two nodes, there may be several connections. When a packet is received, the rate and complexity are updated.

4.1.6 Rate Calculator

The rate of a connection is the number of packets received in the last window. The rate is one factor when considering covert data insertion points. Connections with higher rates are better candidates for blending covert data. Audio and video streams are examples of high rate connections. An example of a low rate connection is a continuous 1-s ping. In addition to the rate of a connection, the complexity of the byte values in a connection are also a key factor.

4.1.7 Complexity Calculator

To calculate complexity, the NBCS allows using either the standard deviation or the probability density function (PDF). In both cases, the complexity of each byte in a connection is calculated, $C_b$. Afterwards, the complexity for the entire connection is calculated as

$$C = \sum_{b=1}^{n} C_b$$

where $n$ is the number of bytes in each packet of connection. The complexity is calculated after each window (see figure 4). A high amount of complexity means that the connection is a good candidate for covert data insertion.
The standard deviation for each byte is calculated using all the values seen for a particular byte position in the last window. The standard deviation method requires relatively little processing and storage, $O(n)$ per connection. A limitation with the standard deviation exists when byte values are discretely distributed (for example, the same five values consistently occur in a byte position). If the discrete values are sparse, then the standard deviation will be high, and, therefore, will be considered as valid covert data insertion points.

Using the PDF (14) for estimating complexity overcomes this problem. The PDF is estimated by keeping a histogram of observed byte values and their frequencies. The optimal case when considering covert data insertion is when all byte values exhibit similar frequencies of occurrence. This results in a uniform distribution. Complexity for each byte is calculated as

$$C_b = \frac{\text{Min}}{\text{Max}}$$  

(2)

where $\text{Min}$ and $\text{Max}$ represent the minimum and maximum byte frequencies in the histogram. If the values are uniformly distributed then the maximum and minimum values will be close in value and the complexity will be close to one. Discrete distributions result in low a complexity value. The PDF comes at a higher performance and storage cost, $O(n^2)$ per connection.

Figure 5 shows sample histograms for three cases. In these cases, the value ranges for the bytes are stored in eight bins (x-axis). Each time a new packet is received, the bin corresponding to the byte value is incremented (y-axis). The leftmost histogram is for a byte that exhibits a
predominate value with some occurrences of surrounding values. The middle histogram shows a byte value that is mostly evenly distributed (which is most favored for covert data placement), while the rightmost graph shows a byte value that has three discrete value ranges.

Figure 5. Sample histograms for different byte frequency distributions.

In implementation, for each byte, a histogram object, which is composed of a hash table, is maintained. The hash key is the byte value and the value pair is the number of occurrences. The histogram can be configured to use different size bins in order to improve memory performance and the probability density function estimate.

4.2 Communication Subsystem

4.2.1 Communications Configuration

The communications configuration parameters in table 2 pertain to the covert insertion point selection and extraction.
Table 2. Configuration parameters for the communications components.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Used By</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate Threshold</td>
<td>Covert Sender/Receiver</td>
<td>The minimum rate (packets per window) that a connection must send data to be a candidate for covert data insertion. A higher value will decrease detectability, but will also reduce throughput.</td>
<td>[1,50]</td>
</tr>
<tr>
<td>Connection Complexity Threshold</td>
<td>Covert Sender/Receiver</td>
<td>This value is calculated by summing the complexity of all bytes in a connection. This value indicates the minimum complexity that a connection must exhibit to be a candidate for covert data insertion.</td>
<td>[1,1024]</td>
</tr>
<tr>
<td>Byte Complexity Threshold</td>
<td>Covert Sender/Receiver</td>
<td>Byte complexity is calculated using the technique described in section 4.1.7. This value indicates the minimum complexity that a byte in a connection must exhibit to be a candidate for covert data insertion.</td>
<td>[0,1]</td>
</tr>
<tr>
<td>Contiguous Complexity Bytes</td>
<td>Covert Sender</td>
<td>Indicates the number of contiguous bytes that must satisfy the Byte Complexity Threshold in order for a connection to be a candidate for covert data insertion.</td>
<td>[Sync Bytes + Checksum Bytes + 1,1024]</td>
</tr>
<tr>
<td>Sync Bytes</td>
<td>Covert Sender/Receiver</td>
<td>Indicates the number of bytes to use for identifying the start of a covert message within a connection’s payload. A higher sync byte count will result in fewer false positives, but may also result in higher detectability and less throughput.</td>
<td>[0,5]</td>
</tr>
<tr>
<td>Checksum Bytes</td>
<td>Covert Sender/Receiver</td>
<td>Indicates the number of bytes to use for calculating the message checksum used to validate covert messages. A higher value will result in fewer false positives, but less throughput.</td>
<td>[0,3]</td>
</tr>
<tr>
<td>Rate To Use</td>
<td>Covert Sender/Receiver</td>
<td>Indicates the percentage of the connection’s rate that will be used for covert data insertion.</td>
<td>[0,1]</td>
</tr>
</tbody>
</table>

As with the analysis configuration, xml files are read and values are stored in a singleton object.

4.2.2 Covert Sender

The covert sender takes input data from a human communicator (user) and chooses insertion points. Connections with high rates and complexities are considered for covert data insertion. Within these connections, bytes that exhibit high complexities are replaced with covert data and the new modified packets are injected into the network.

The covert sender component contains a thread that accepts input from a user. When new input is received, the data are placed in a data buffer. After each window, connections with high data rates and high complexity are selected.

These connections are then further filtered by keeping only those with sufficiently long sequences of bytes with high complexity. For each of these connections, a copy of the last packet received along with the longest sequence start and end points are retrieved. Next, bytes in the
buffer are used to replace the sequence of bytes in the packet (shown in figure 6). The modified packets are then sent using the jNetPcap API.

![Figure 6. Covert data insertion process.](image)

Covert data (blue) are appended with a header and footer (brown) and replaces the byte series. For each window, only one covert packet is sent per connection. Also, a penalty parameter can be set in the configuration, to ensure the same connection is not used for a certain number of windows.

### 4.2.3 Covert Receiver

The covert receiver decodes hidden data within connection packets. In an ideal situation, a covert receiver would monitor the same traffic as a covert sender and each would come to the same conclusion for covert insertion points. However, in practice, even if two entities are connected to the same hub, the same traffic is not received simultaneously. For this reason, the following header and footer fields are included in covert messages in order to improve communication reliability.

- **Synchronization bytes**—these bytes are placed before any covert data. A receiver can search a packet payload for these synchronization bytes as a first indication of possible covert data. These bytes can also be used as an identifier for the sender when multiple communicators are present.

- **Message length**—This parameter indicates the length of the covert data within a packet payload.

- **Sequence number**—This optional parameter is placed after the synchronization bytes and can be used to identify covert data loss. Data loss may be common depending on the
protocol used to send covert data, e.g., UDP is unreliable and lost packets are not retransmitted.

- **Message checksum**—This optional parameter reduces false positives by summing all of the byte values in the covert payload. If the receiver observes a correct checksum, then the message is considered valid.

In the most conservative case, a covert receiver will search all incoming packets on all connections for covert data. To improve performance, the covert receiver may also employ a selective search. Instead of searching all messages, the receiver will only search messages with a certain rate and complexity. However, this may also increase covert data loss, because of the asynchronous nature of packet arrivals. The configuration component allows covert communicators to specify values to balance this trade-offs.

### 4.3 Graphical Subsystem

The graphical subsystem composes the GUI that a covert communicator may use to analyze the state of the network. The GUI may also be used for text communication and to send files unreliably. However, there is a performance cost associated with the display windows. Figure 7 shows a screenshot of the connections display and the complexity display.
4.3.1 Statistics Display

This component uses the JFreeChart API to display byte statistics from the connections. Statistics that are displayed include the total throughput during each window. For every 10 windows the average, max, and minimum throughput is displayed. A screenshot of the statistics display is shown in figure 8.
This display allows a communicator to determine configuration values to use for blending depending on the amount of traffic observed. Also, especially in a wireless network, it will help to determine whether covert communicators observe similar amounts traffic, it may be that packets are lost due to proximity, environment, and others.

In implementation, a JFreeChart is created that contains four series, one for each statistic. After each window, when a connection calculates the new counts, an event is triggered, and the display updates.

### 4.3.2 Connections Display

The connections are displayed in a sortable table that contains the header fields of all packets in each connection along with the rate and connection complexity. Double-clicking on a connection will open the byte complexity display.

Display data are provided from the connection objects when they are updated using an event-driven update mechanism (specifically, by extending the AbstractTableModel class).
connections display is a chat panel for communicators to send and receive covert messages and a file panel to send and receive files.

### 4.3.3 Byte Complexity Display

The byte complexity display allows communicators to view the complexity of each byte in a connection in a sortable table. A communicator can use the complexity display to determine which bytes in a connection contain higher complexity and can also aid in determining optimal bin sizes for histograms. For example, larger bin sizes may cause high complexity values for the following reason. Byte values will have fewer bins to hash into. Bins will cover a wider range of values and it may be the case that even with sparse byte values, frequencies may seem uniform.

---

## 5. Evaluation

In order to test the performance of the NBCS, we implemented a tool and tested it in real network environment. The tool is written in the Java programming language and uses the open source JNetPcap library for network operations.

### 5.1 Experimental Setup

We evaluated our covert method by measuring throughput, reliability, and observable detectability under different network loads. We evaluate reliability by measuring packet loss and then we evaluate trade-offs between throughput and detectability by testing the method using different parameters values. For the first experiment, the network setup contains eight nodes connected on a 100 Mbps network hub. One node is the covert sender, one node is the covert receiver and the other six nodes are overt communicators that exchange mp3 audio using the real-time service protocol (RTSP) enclosed in UDP.

In total there are three pairs of connections among the six nodes. Each node acts as a server and client to a neighboring node (i.e., node A serves node B and node B also serves node A, etc.). The publicly available live555 media server software is used to serve the audio and the video lan client (VLC) tool is used to play the audio. Each node serves a different mp3 file (a total of six different mp3 files are used). The audio files are set to loop in order to keep a continuous data stream. The mp3 audio files are recorded conversations taken from a dialog corpus (13). One node was configured to run the Snort IDS in order to determine whether the covert communication method could be detected. The rule set included the most recent anomaly and signature-based rules available from the Snort web site and custom rules aimed at detecting covert communication in header, footer and unused message fields.

In order to observe the effects of the network load on throughput (specifically packet loss), we conducted a second experiment with a higher amount of network congestion. For the second experiment, the network setup consisted of the same eight nodes, plus six additional nodes
exchanging TCP packets. We used a traffic generator to generate the TCP traffic, which was mostly audio and video data.

Lastly, a third experiment was executed to determine capabilities in a wireless environment. This experiment was a duplicate of experiment 1, except that instead of using a network hub, we used an 802.11 g (54 Mbps) wireless router with nodes in an enclosed 12-ft ×17-ft area. We ran the experiment with nodes wireless cards in both promiscuous and monitor mode.

Table 3 shows the network load conditions during both experiments.

<table>
<thead>
<tr>
<th>Load A</th>
<th>Load B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overt Nodes</td>
<td>6</td>
</tr>
<tr>
<td>Packets/sec</td>
<td>80–100</td>
</tr>
<tr>
<td>Bytes/sec</td>
<td>95,000–115,000</td>
</tr>
<tr>
<td>Active Connections</td>
<td>15–20</td>
</tr>
</tbody>
</table>

In all experiments, only the UDP connections were used as carriers for covert communication. The reason UDP was used is because the protocol is connectionless and there is no control mechanism as there is in TCP (with sequence numbers).

During the send phase, only payload data are modified, not header fields. Duplicating TCP messages could alert an adversary to the covert communication. One way that TCP could be used is if a covert sender is also a legitimate sender that embeds covert data in the payload of select messages. As the experimental control, the following parameter values were used. We used two sync bytes and two checksum bytes to virtually eliminate false positives. The other parameters were chosen favoring detectability over throughput.

- Window Size = 1000 ms
- Sync Bytes = 2
- Checksum Bytes = 2
- Rate to Use = 0.1
- Protocol to Use = UDP
- Rate Threshold = 10
- Connection Randomness Threshold = 10
- Contiguous Random Byte = (1+3) + 1 = 4
We ran each experiment with varying values for the Byte Randomness Threshold parameter and we measured the throughput and reliability during each run.

For each run, we first started a covert sender, which would automatically begin sending covert packets when valid threshold values were observed. Each packet included a sequence number (0–255) as the first byte of data. The sequence numbers were used to identify lost packets. On the receiver, missing or out-of-order sequences indicated missing packets.

On the sender, a large file was loaded into the covert data buffer. As a result, the buffer always contained enough data to fill all candidate carriers during each window. This was done to measure maximum throughput.

The receiver was started 15 s after the covert sender. Packet loss was observed only after the first packet was received. The receiver was run for 5 min, during which all incoming covert messages were logged. After the experiment, we analyzed the logs to measured throughput and packet loss.

6. Results

We measured throughput using different Byte Randomness Threshold values (to represent detectability levels). Figure 9 shows these results. The amount of throughput is consistent under both network loads (because the same UDP traffic was present for both). Also, as the threshold value is increased, the throughput decreases, but this also means that the data are placed in areas with higher randomness, which would be more difficult for an adversary to find. It should be noted that when using a value of 0.9 for the Byte Randomness Threshold, no covert data were sent. Although there were some individual bytes that exhibited this high randomness, there were not enough contiguous bytes to fit the covert data.

When using promiscuous mode, roughly 50% of the packets were lost. The results presented here are from using monitor mode. We achieved results similar to when using the network hub, except with a lower throughput. This is probably due to the bandwidth limitations of our wireless router. An interesting note is that when in monitor mode, roughly 10 duplicates per covert message (duplicates are not counted for the results) were seen by the receiver. This is probably due to redundancy settings by the manufacturer on the wireless router.
We measured reliability by counting the misses associated with the throughput presented in figure 9. Figure 10 shows these results. The data points are the total number of misses averaged over the 5-min runs. Even using UDP connections, which are inherently unreliable, the miss counts are minimal compared to the available throughput received under both network loads. The reason that the packet loss during the higher network load is not monotonically increasing is attributed to the bursty nature of some of the TCP communication that was present. Since the number of misses were small, it may be feasible for a receiver to request resubmission of lost packets. Also, this shows that it may be favorable to communicate covertly over congested networks to decrease detectability because the communication is still reasonably reliable.
To determine the utility of the NBCS with the given network, we measured the number of packets that were sent during both network loads (figure 11). As mentioned earlier, the window size used during the experiments was 1 s. Also a maximum of six connections were considered as candidate carriers (only the six satisfied the parameters). There were other UDP connections used for controlling the media sessions, but only the connections with audio data had suitable randomness values. As a result, the maximum number of packets that were sent each second was six. This is observed in the graph. A high degree of randomness is expected because the packets contain speech audio. When the threshold parameter values are higher, the number of packets used decreases. This is a good indication that the audio data differed. Some connections had higher levels of randomness due to the fact that in some dialogs, the speakers were more verbal. If the speakers talked less, there was more silence, which translated to less randomness. Although the same connections were used as covert carriers during both network loads, there are slight differences in the values. This is because the sender and receiver were started while background traffic was already running. The randomness threshold were reached at different times depending on the network traffic present during the instantiation of the sender and receiver.
Regarding the IDS, IP header checksum field error alerts were triggered, which are fixed by recalculating checksums on the covert messages, but no other alerts were raised.

7. Conclusions and Future Work

We have developed a novel method that can be used by threat computer network operations (CNO) tools to coordinate in a distributed fashion. Evaluation of the method shows that the communication works with limited data loss and does not alert an IDS. We have also demonstrated how the method can be configured to balance between throughput and detectability.

In the short-term, we will determine the effects of higher throughput on quality of service (e.g., distortion, delay, etc.). In addition, we will investigate whether it is more beneficial to hide covert data based on byte similarity as in (13). Lastly we will look into automatic methods for tuning the parameters in table 1 and table 2 in real time depending on network characteristics.

We plan to propose the continuation of this work to transition into TTM programs. Along with the short-term work mentioned above, there are several theoretical research projects that may arise. Some examples include the use of lower-layer protocols, e.g., physical waveforms for covert communication or for identifying common patterns that may potentially available to
adversaries. Another avenue is determining whether the use of this method provides better way for detecting anomalies and implementing better intrusion detection systems. Lastly, the method may be useful for creating secure systems to hide node endpoints and traffic behind decoy traffic.
8. References


## List of Symbols, Abbreviations, and Acronyms

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>ARL</td>
<td>U.S. Army Research Laboratory</td>
</tr>
<tr>
<td>ARP</td>
<td>address resolution protocol</td>
</tr>
<tr>
<td>CND</td>
<td>computer network defense</td>
</tr>
<tr>
<td>CNO</td>
<td>computer network operations</td>
</tr>
<tr>
<td>COTS</td>
<td>commercial-off-the-shelf</td>
</tr>
<tr>
<td>DNS</td>
<td>domain name system</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>hypertext transfer protocol</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet control message protocol</td>
</tr>
<tr>
<td>IDS</td>
<td>intrusion detection system</td>
</tr>
<tr>
<td>IP</td>
<td>Internet protocol</td>
</tr>
<tr>
<td>LACK</td>
<td>Lost Audio PaCKets Steganography</td>
</tr>
<tr>
<td>LSB</td>
<td>least significant bit</td>
</tr>
<tr>
<td>MAC</td>
<td>media access control</td>
</tr>
<tr>
<td>NBCS</td>
<td>network blending communication system</td>
</tr>
<tr>
<td>PBA</td>
<td>polymorphic blending attack</td>
</tr>
<tr>
<td>PDF</td>
<td>probability density function</td>
</tr>
<tr>
<td>RTSP</td>
<td>real-time service protocol</td>
</tr>
<tr>
<td>SLAD</td>
<td>Survivablity/Lethalilty Analysis Directorate</td>
</tr>
<tr>
<td>TCP</td>
<td>transmission control protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>user datagram protocol</td>
</tr>
<tr>
<td>VLC</td>
<td>video lan client</td>
</tr>
<tr>
<td>VoIP</td>
<td>voice over Internet protocol</td>
</tr>
<tr>
<td>No. of Copies</td>
<td>Organization</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>1 (PDF)</td>
<td>ADMINSTR DEFNS TECHL INFO CTR DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218</td>
</tr>
<tr>
<td>3 HCs</td>
<td>US ARMY RSRCH LAB ATTN RDRL CIO LT TECHL PUB ATTN RDRL CIO LL TECHL LIB ATTN IMNE ALC HRR MAIL &amp; RECORDS MGMT 2800 POWDER MILL ROAD ADELPHI MD 20783-1197</td>
</tr>
<tr>
<td>1 CD</td>
<td>US ARMY RSRCH LAB ATTN RDRL SLE E D NEVAREZ BLDG 1624 ROOM 225 WSMR NM 88002-5513</td>
</tr>
<tr>
<td>1 WORD VERSION</td>
<td>MELE ASSOCIATES INC ATTN RDRL SLE I D NEVAREZ BLDG 1624 ROOM 225 WSMR NM 88002-5513</td>
</tr>
<tr>
<td>1 CD</td>
<td>US ARMY RSRCH LAB ATTN RDRL SLE I J ACOSTA WSMR NM 88002-5513</td>
</tr>
</tbody>
</table>

Total: 8 (1 PDF, 4 HCs, 2 CDs, 1 Word Version)