Lightweight Network Steganography for Distributed Electronic Warfare System Communications

Tim Lei
Department of Computer Science
San Francisco State University
1600 Holloway Avenue
Thornton Hall 906
San Francisco, CA 94132
tlei0929@gmail.com

Jeremy Straub ***(Contact Author)***
Institute for Cyber Security Education and Research
North Dakota State University
1320 Albrecht Blvd., Room 258
Fargo, ND 58108
jeremy.straub@ndsu.edu

Benjamin Bernard
Department of Computer Science
North Dakota State University
1320 Albrecht Blvd., Room 258
Fargo, ND 58108
ben.bernard@ndsu.edu

Abstract

This paper presents the application of a modified implementation of the StegBlocks TCP method as part of the Distributed Electronic Warfare System. The existing system is not equipped with a secure information communications mechanism for transmission between assimilated hosts and from hosts back to the server. The method implemented utilizes network steganography to provide covert data transmission through the network using network packets. The proposed implementation is compared to another implementation of the same method on the aspects of the implementations' usability, versatility, and applicability. Discussion on how the proposed implementation is more suitable than the alternate implementation is presented. Future proposed improvements to the implementation are also discussed.

Keywords: information communications and security, Distributed Electronic Warfare System, network steganography, StegBlocks TCP method

Submission Type: Short Research Paper

1. Introduction

Cyberwarfare is a modern form of warfare which transforms battles from occurring on the physical ground to battles across the virtual grounds of computer networks. It is the next generation of warfare [1] and it allows battles to be fought prospectively without actual human bloodshed. In physical warfare, humans with weaponry physically battle on a battlefield. In cyberwarfare, on the other hand, humans utilize computers as cyber-weapons to remotely and virtually battle in a virtual battleground. Even though the physical damage inflicted can be reduced, significant damage can be inflicted upon both data and real-

world assets. These damages can be extremely destructive to societal infrastructures. As cyberwar becomes more prominent, information security requires growing attention.

Cyberwarfare has been an ongoing, to various extents, between global powers, including among the United States and Russia. Cyberwarfare can be a war of information. Whichever force has access to more information and can exploit this information against an opponent or can defend its information from being accessed by opponent forces gains an advantage in cyberwarfare. It is essential to be on the side with an advantage in cyber warfare as it provides opportunities to act advantageously, either offensively or defensively. Once one side has an advantage over the other, the trend is very likely to continue. In many cases, the side with disadvantage can only react to the actions from the advantaged side to neutralize possible damages.

Modern cyberwarfare, which utilizes attack systems, is growing rapidly. The development of artificial intelligence is growing at a swift rate. New technologies incorporate the power of artificial intelligence into their products to boost their performance. The application of artificial intelligence is expanding to more and more fields, and there is no doubt that cyberwarfare attack systems are part of this trend [2]. Attack systems that use artificial intelligence use these autonomous capabilities to greatly improve the functionality of the attack system.

Distributed AI systems require methods of communications. In some cases, there is significant benefit to this transmission being covert. Passing information through the internet is simple; however, sending or receiving information through the internet without being discovered by opponent forces is not. The Distributed Electronic Warfare System (DEWS) requires communications between the central blackboard and local blackboards to update them with the knowledge that has been gathered. This information is transmitted through the network, which raises security concerns. The information transmitted between central and local blackboards is sensitive and, in particular, a way to securely transmit information through the network without being detected is required. There is currently no implementation of an information security method for this purpose in DEWS, which makes the system insecure and vulnerable to attacks. A network steganography method, based on the StegBlocks TCP method [3], can be applied to the DEWS to ensure the covert transmission of information within the DEWS.

The StegBlocks TCP method is a form of steganography which involves hiding data inside multimedia data or network data. Network steganography uses a network as cover media to hide data and transmit data without being detected. This paper presents the implementation of a network steganography StegBlocks TCP-derived method for the DEWS to secure the information transmitted between existing hosts and the assimilated machines. The clients-server implementation (CSI) is used to improve the security of communication between machines. The implementation presented in this paper is compared to another implementation [4] of the same method on the metrics of usability, versatility, and applicability of the method.

2. Background

This section reviews prior work, in several areas, that provides a foundation for the current work. First, prior work on the Blackboard Architecture is reviewed. Next, prior work on a distributed electronic warfare system, based on the Blackboard Architecture is presented. Then, steganography is discussed. Finally, prior work on network steganography and the StegBlocks TCP method is covered.

2.1. Blackboard Architecture

The Blackboard Architecture [5] takes the concept of an expert system and transforms it into a task solving architecture with three main components: the blackboard, knowledge sources and the control.

Similar to the expert system, which passively infers a possible solution, the Blackboard Architecture works on a task or a goal to generate solutions or partial solutions. Enhancements have been made on the Blackboard Architecture concept proposed by Hayes-Roth [6]. The Blackboard Architecture adds a layer of control to exploit the capabilities of AI systems and to adapt to the changing environment which may include newly generated tasks or partially known solutions. A distributed Blackboard Architecture implements the Blackboard Architecture with a host/central blackboard which generates a hierarchical tree of local blackboards and the central blackboard distributes tasks to the local blackboards to solve. After a partial or the whole solution is accomplished, the solution is sent back to the central blackboard knowledge base, where all the information and tasks are stored.

2.2. Distributed Electronic Warfare System

The DEWS was developed based on the aforementioned distributed Blackboard Architecture [8], as shown in Figure 1. The DEWS [7] has been proposed to be used in cyberwarfare against other forces to control opponents' complex networks and computing systems. The proposed system implemented artificial intelligence to gather information about targets, to make the decision on suitable methods to exploit to attack these targets, to launch payloads, and to propagate deeper into the adversary's connected network of computers. The DEWS is based on a central blackboard and local blackboards hosted on both command stations and assimilated machines. Each local blackboard contains a portion of the central blackboard's knowledge, based on the local machine's capabilities, location and the logistics of data transfer. The communications between the central blackboard and local blackboards is crucial and the information sent between them needs to be secured, covert and traceless. At present, DEWS has not incorporated information security or covertness in the communications between the blackboards.

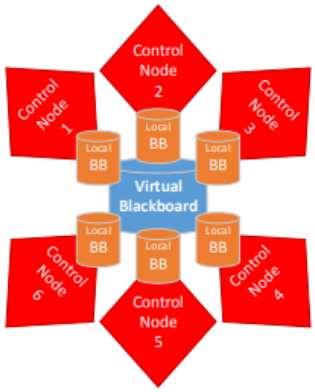

Figure 1. Blackboard-Based Electronic Warfare System [8].

2.3. Steganography

Steganography is defined as "the art or practice of concealing a message, image, or file within another message, image, or file" [9]. It has been used with a variety of forms of media. Network steganography is gaining its importance from its powerful capability to transmit secret data through a network. There are four main attributes of network steganography communications [3]. Each is now briefly discussed. Bandwidth is the amount of data can be handled at once. Undetectability is the extent to which the hidden message is undetectable and untraceable. Robustness characterizes the integrity of the hidden message after the carrier media is altered or damaged. Finally, cost refers to the level of distortion on the message carrier media caused by the steganography method.

There are many forms of steganography. These include text, image, voice/video, and network steganography methods. Steganography in text hides a secret message within a block of text using an

encoding scheme. Image steganography hides a secret message in an image file by modifying certain bits within the image. Voice/video steganography is similar to image steganography and uses similar techniques to hide a secret message within the bits of the voice or video datagrams. Network steganography hides secret messages in network traffic.

2.4. Network Steganography

Most steganography studies were conducted on text, image, voice/video steganography as opposed to on network steganography. However, a number of methods have been developed for network steganography. These include StegBlocks [3], PadSteg [9], HICCUPS [11], RSTEG [12], WiPad [13] and ReLACK [14]. These methods are based on different layers of the Open Systems Interconnection Reference Model [15].

2.5. StegBlocks TCP Method

In the StegBlocks TCP method [3], which the CSI method proposed herein is based on, TCP connections are established between two machines. The two select a steganographic key and blocks are identified. Each block has a value which is based on "the last x bits of the number of TCP segments" in it [3]. If this block does not have the desired value for the message that will be transmitted with it, it must be changed to have this value.

3. Implementation

The DEWS uses network traffic to transfer data information; however, sending data through the network without disguise can be easily tracked and intercepted by cybersecurity professionals or detected by programs searching for intrusions or anomalies. Network steganography helps to obfuscate the transmission of data that systems receive or send to, from, or between assimilated machines through the network. In the DEWS, agreements exist between the nodes with local blackboards that comprise the virtual central blackboard. Only the machines with applicable agreements can readily discover the hidden data that is transferred.

The work presented herein uses the conceptual model of the StegBlocks TCP method, with some changes. These changes, in particular, are designed to minimize the computing overhead if the protocol to facilitate its use on computationally limited devices, such as Internet of things devices.

Under the method used herein (the CSI), like with the StegBlocks TCP method, connections are established between two nodes. Among these TCP connections, two gatekeeper connections are selected and the rest of the connections are treated as connections to transport payload data. The functionality of the connections depends on the ports they are connected to. The first port, port A, of the sender's connection is selected as the first gatekeeper. The last port, port Z, of the sender's connection is selected as the second gatekeeper. The rest of the ports, ports B to Y, of the sender's connection are selected for transmission, based on the value of the data. The ports are selected randomly.

Using CSI to send the word 'cat' with ASCII encoding, the values for the characters are 99, 97 and 116. The total number of packets required to send the word is the sum of its ASCII values plus two packets for the gatekeepers, which is 318 packets in total. Figure 2 demonstrates this for sending the word 'cat' with this method.

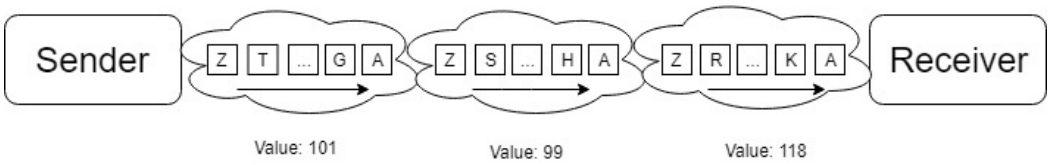

Figure 2. Example of sending the word 'cat'.

3.1. Encoding Scheme

When using the ASCII value encoding to send characters, the number of packets required to represent a character can be as high as 127. It is, thus, not always efficient to send a character based on the character's ASCII decimal value. To optimize efficiency, the method of sending a character is modified from sending a number of packets based on its ASCII value to a number based on the digits of the ASCII value. For example, the character 'A' has an ASCII value of 65. By applying the new method, the value becomes '065' and the number of packets required to represent the character is 0 + 6 + 5 = 11. By including two packets for the two gatekeeper ports to send each digit across, number of packets required is 11 + 6 (2 packets for gatekeepers * 3 digits) = 17. By converting to this new method, the largest number of packets required is 18 + 6 = 24 for the character 'c' with an ASCII value of 99. This can be compared to the unmodified method, where the largest number of packets used is 129 packets. Given this, the modified version can be up to five times as efficient than the old method.

Table 1. Encoding table of three different digits of character values.

Character	ASCII Decimal Value Coding	New Value Coding	Packets Needed for New Coding	Total Packets Needed
TAB	9	'009'	0 + 0 + 9 = 9	9 + 6 = 15
K	75	'075'	0 + 7 + 5 = 12	12 + 6 = 18
d	100	'100'	1 + 0 + 0 = 1	1 + 6 = 7

3.2. Scenario

A simulation of multiple clients communicating with a server was implemented to simulate data transfer between nodes in the DEWS using the CSI method. A central server was implemented to simulate the Blackboard command station and clients were implemented to simulate the assimilated and other geographically diverse nodes controlled by the DEWS. The functions of the server node are to listen for incoming packets, decode the carried message from the number of packets sent to the server, and store the message into a text file for future use. The functions of the clients are to capture packets, store them in a pool of packets, modify the headers of the packets, and send the modified packets through a predefined set of ports using a predefined encoding scheme. The server acknowledges the encoding scheme used by the clients. Both the server and clients use the Scapy tool in their implementation. The data in the text file that was used in this scenario is 10kB.

3.3. Client-Side Implementation

The client-side of the implementation has three clients which use the same interface. Three clients connect to different sets of ten open server ports. These ports were pre-selected and assigned to each client. Two of the open ports were used as gatekeeper ports, one was used to tell the server when the end of information transmission was reached. The rest of the ports were used as transport ports. The encoding scheme applied to the message was converting each character to its ASCII value, converting this value to its three-digit integer string, and filling empty digits with zeros. Additionally, the software modifies the source IP address to the IP address of the client and the destination IP address to the server's IP address, in the packet's IP header. It also modified the destination port in the packet's TCP header, and sent the

packet. This process of modifying and sending packets was repeated until the number of packets required for each digit were sent. For testing, each client opens a text file containing a message to be transferred. It then reads a character, calculates the number of packets required to send the character, reads in the requisite number of packets from the pool of packets and then sends each character using the aforementioned encoding scheme.

3.4. Server-Side Implementation

The server software is equipped with a sniffer to listen on its open ports assigned to the three clients and receive packets from the three clients. There are three counters, one for each set of ports, to count how many packets are sent through transport ports of the three clients. The count is converted to a string representation of the ASCII value of the character being sent. Once the string has all three digits, the string is converted to an integer value and converted to an ASCII character. The decoded character is then written into the respective resulting text file for the particular client. Once all three client finish their activities, the server stops listening on the ports and is shutdown.

4. Data Collection

To evaluate the performance of this approach, a script was written to run both the server and the clients. Network packets were pre-captured through the Ethernet port and imported into the program for the use of the CSI method to transfer hidden messages. The implementation of the method in [3] (client-server method) was also evaluated. This method (from [3]) uses a single server to send data to a single client rather than having a server send data to multiple clients. In both scenarios, the same text file is used.

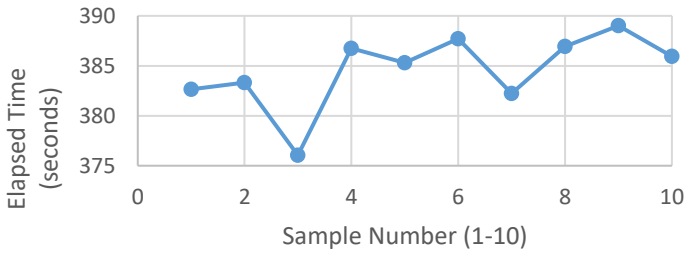

Figure 3. Sending sample text 10 times from three clients to a server.

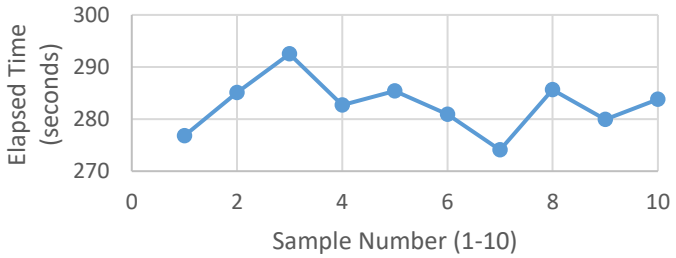

Figure 4. Sending sample text 10 times from a client to a server.

The average elapsed time for transmission for three clients sending to a server (clients-server method) is 384.59 seconds with 111,738 packets required to send the text file. The average elapsed time for the client-server method was 282.68 seconds with 86,176 packets required to send the text file. It is notable that the client-server program skips characters which are not in its lossy collection, which only includes lower case letters and the space character. On the other hand, the CSI can transfer all ASCII characters.

The byte rate for the client-server program, considering that there are three clients, is 77.57 bytes per second. The byte rate for clients-server program is 25.86 bytes per second. Test run results are shown in Figures 3 and 4.

5. Assessment

This section assesses the CSI (which is based on the StegBlocks method), in terms of multiple criteria. The criteria considered include usability, versatility, applicability and undetectability. Each is now discussed.

5.1. Usability

The CSI, based on the StegBlocks method, was implemented with multiple clients to allow its use for DEWS communications between hosts. The DEWS is a highly interconnected network where multiple instances of multiple-to-one client-server structures are required. Thus, a single client to a single server communications paradigm would not be appropriate for this application. The multiple-to-one client-server structure is needed for the system to operate with its full capabilities. Once the attack system breaches an adversary's machine and takes control of the machine, the CSI begins to operate by gathering the required IP address and the ports of the machine. It also collects the information for use by the attack system for the purpose of further propagation of the infection of machines or leaving behind backdoors for future access.

5.2. Versatility

The encoding scheme used in the CSI can transmit all ASCII characters while the implementation in [3] can only transmit lowercase letters and the space character. The CSI made it possible for uppercase letters, numbers, punctuation, and special characters to be transferred. The more limited implementation of the StegBlocks method would not be useful for some applications if the only data which can be transported is lowercase letters and the space character. For example, if a webpage link needed to be transmitted, the link would contain numbers and the special character '/' in addition to letters.

5.3. Applicability

The implementation of the client-server structure of the StegBlocks method and its derivatives can be applied to other future attack systems, in addition to DEWS, which utilize the network to communicate. Many attack systems may need the capability for transmitting secret data from multiple assimilated machines back to a central point of command. In such a case, the implementation of a single client to a single server would require excessive management on numerous individual machines.

5.4 Undetectability

The level of undetectability is the core requirement for steganographic methods. The CSI did not use the Vernam cipher as suggested in [3] for perfect undetectability. The clients-server implementation, instead, focuses on the transfer of overt text to support covert communication between multiple hosts. Perfect undetectability is not provided by the CSI because there is the possibility of detection using statistical analysis of the number of packets sent. Analysis may reveal the order of digits and the payload. The CSI does not include a mechanism which alters the packets' payload, thus the analysis of the contents of packets has no effect and does not increase the probability of triggering detection.

5. Conclusions & Future Work

In this paper, a modified lightweight implementation of the StegBlocks method was presented and implemented for evaluation for possible incorporation into the DEWS. Comparing the two, the client-server implementation did not outperform the clients-server implementation. The CSI has a multiple-to-one client-server structure and is able to transfer any ASCII characters while the client-server implementation has a one-to-one client-server structure and is only able to transfer lowercase letters and the space character.

While the implementation shows promise for future use, a number of enhancements are needed as future work. First, the program should be optimized to run in the background. The implementation is more useful for the DEWS if it can run covertly. The CSI did not implement a mechanism where the program can be run in the background without being detected. This improvement could hide the activity of the program and delete all traces of it after the program finishes running.

Another prospective improvement is to randomize port selection. The proposed implementation uses pre-assigned ports for both the gatekeeper and transmission ports. This impairs the level of undetectability if statistical analysis is run on the system. The process of randomly choosing ports and repeating the selection process frequently can greatly reduce the chance of being detected using statistical analysis. It is planned that these improvements could be implemented as future work.

Acknowledgements

This research was supported by the United States National Science Foundation (Award # 1757659). Some facilities and equipment were provided by the NDSU Institute for Cyber Security Education and Research and the NDSU Department of Computer Science.

References

[1] T. Franz, “The cyber warfare professional: realizations for developing the next generation,” *Air Sp. Power J.*, vol. 25, no. 2, 2011.
[2] J. Straub, “Artificial intelligence is the weapon of the next Cold War,” *The Conversation*, 29-Jan-2018.
[3] W. Fraczek and K. Szczypiorski, “Perfect undetectability of network steganography,” *Secur. Commun. Networks*, no. April, pp. 2998–3010, 2014.
[4] P. Bak, J. Bieniasz, M. Krzeminski, and K. Szczypiorski, “Application of perfectly undetectable network steganography method for malware hidden communication,” *2018 4th Int. Conf. Front. Signal Process. ICFSP 2018*, pp. 34–38, 2018.
[5] H. P. Nii, “Blackboard systems: Part I,” *AI Mag.*, vol. 7, no. 3, pp. 38–53, 1986.
[6] B. Hayes-Roth, “A blackboard architecture for control,” *Artif. Intell.*, vol. 26, no. 3, pp. 251–321, 1985.
[7] J. Straub, “Blackboard-based electronic warfare system,” in *Proceedings of the ACM Conference on Computer and Communications Security*, 2015, vol. 2015-Octob.
[8] I. Burton and J. Straub, “Autonomous Distributed Electronic Warfare System of Systems,” *2019 14th Annu. Conf. Syst. Syst. Eng.*, pp. 363–368, 2019.
[9] “Definition of Steganography,” *Merriam-Webster Dictionary*, 2020. [Online]. Available: https://www.merriam-webster.com/dictionary/steganography. [Accessed: 08-Jun-2020].
[10] B. Jankowski, W. Mazurczyk, and K. Szczypiorski, “PadSteg: Introducing inter-protocol steganography,” *Telecommun. Syst.*, vol. 52, no. 2, pp. 1101–1111, 2013.
[11] J. P. Black *et al.*, “Steganography in TCP / IP Networks . Outline,” *Proc. - 2010 2nd Int. Conf. Multimed. Inf. Netw. Secur. MINES 2010*, vol. 4, no. 3, pp. 225–229, 2014.
[12] W. Mazurczyk, M. Smolarczyk, and K. Szczypiorski, “Retransmission steganography and its detection,” *Soft Comput.*, vol. 15, no. 3, pp. 505–515, 2011.
[13] K. Szczypiorski and W. Mazurczyk, “Steganography in IEEE 802.11 OFDM symbols,” *Secur. Commun. Networks*, no. March 2011, pp. 118–129, 2014.
[14] M. Hamdaqa and L. Tahvildari, “RELACK: A reliable VoIP steganography approach,” *Proc. - 2011 5th Int. Conf. Secur. Softw. Integr. Reliab. Improv. SSIRI 2011*, pp. 189–197, 2011.
[15] J. Lubacz, W. Mazurczyk, and K. Szczypiorski, “Principles and overview of network steganography,” *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 225–229, 2014.