Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

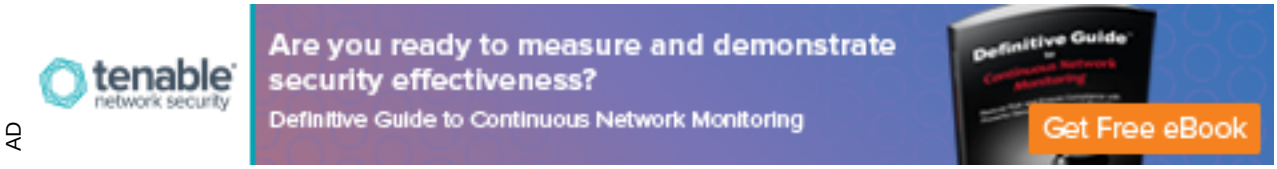
This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Network Covert Channels: Subversive Secrecy

Confidentiality, one of the primary goals of information security, is most frequently achieved utilizing cryptography. There exist other methods to conceal information from unauthorized or undesired viewers. One category of noncryptographic concealment techniques, called covert channels, can hide information locally on a system or mask traffic as it passes over a network. This report will explore the art of hiding data as it traverses a network. This paper will analyze motivations for network covert channels, survey cu...

Copyright SANS Institute
Author Retains Full Rights

AD



Network Covert Channels: Subversive Secrecy

Raymond Sbrusch
sbrusch@gmail.com

Abstract: *Confidentiality, one of the primary goals of information security, is most frequently achieved utilizing cryptography. There exist other methods to conceal information from unauthorized or undesired viewers. One category of non-cryptographic concealment techniques, called covert channels, can hide information locally on a system or mask traffic as it passes over a network. This report will explore the art of hiding data as it traverses a network. This paper will analyze motivations for network covert channels, survey current data hiding research and tools, and evaluate countermeasures to detect and prevent network-based steganography.*

1. Introduction

Steganography is the practice of concealing information in channels that superficially appear benign. The National Institute of Standards and Technology defines a covert channel as "any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy" [13]. In other words, a network covert channel is manipulation of a communication protocol to transfer information in a way outside the protocol's specification. In part, these channels exist because of undefined, optional, or permissive values allowed to be assigned to certain fields. Steven Bellovin summarizes the issue in stating that "we are not concerned with flaws in particular implementations of the protocols … rather, we discuss generic problems with the protocols themselves" [5]. This paper will analyze motivations for using covert channels, define criteria used when evaluating covert channels, survey existing research and tools, and review mechanisms for detecting and preventing network-based steganography.

2. Motivations

Covert network channels provide an alternative, somewhat subversive means of achieving confidentiality and maintaining anonymity. There exist both illicit and honorable motivations for using network steganography. For one, many countries maintain strict regulations over use of cryptography [11]. It is possible for citizens of some countries to be imprisoned if they encrypt messages and attempt to evade government detection. Even in the United States, the federal government expressed interest in key escrow systems that would enable trusted agencies to decipher encrypted messages [8]. In some cases, the contents of the message are less meaningful to observers than the mere fact that two parties are communicating. If the observer witnesses encrypted communications, they may become curious about the motivation for hiding something. In this case, the best way for a message sender and recipient to avoid traffic analysis is to never communicate directly. Some covert channel techniques described in this report can be used by benevolent entities that wish to watermark communications from opposition organizations. For example, a country can digitally watermark a network datagram as it leaves the network of one enemy. They can observe datagrams entering other organizations network and thus determine who cooperates with whom.

Covert channel tools are not attack tools; they are not used to break into systems. In some scenarios, they can be used alongside an exploit tool to leech information off a system or to secretly deliver commands to a system. For example, an attacker can compromise the web server of a popular banking site. The usernames and password of the bank's customers are much more valuable than simply defacing the web page. The attacker can plant a password sniffing program on the site and use a covert channel tool to stealthily deliver credentials from the web server to another location. Messages can be delivered on a scheduled interval, such as with a cron job, or they can be sent interactively, as with an email system. The scenario works in reverse, as well. The covert channel tool can be used to retrieve commands from an external site that are to be run on the compromised, internal server.

3. Evaluation Methodology.

The criteria for evaluating covert channels are not inherently different from those used when evaluating open communication mechanisms, except in a few cases. Bandwidth, ease of implementation, and range are still important traits. However, the ways they are measured may differ. Additional criteria include permissibility, probability of detection, and level of anonymity. Bandwidth can be measured in term of bits per packet and bits per session. For example, some systems encode a character in every frame that leaves a network. Other systems, such as those based on TCP Initial Sequence Numbers, may only encode one character in each TCP session.

© SANS Institute 2006, Author retains full rights.

Application layer based channels may deliver large quantities of information in a single segment. Implementation factors include whether specialized hardware is required, operating system modules need to be modified, or control over specific systems in the network path is required. As with the OSI layer in general, some protocols have limited range. For example, Ethernet frames can not easily be tunneled across the Internet to remote hosts. Similarly, covert channels based modifications of the Ethernet specification have limited range as well. Even network layer covert channels face unique obstacles. For example, many protocols are not permitted to travel outside a firewall and leave a protected network. Protocol headers may be modified or carefully inspected while the packet is in transit. For example, Network Address Translation (NAT) typically replaces the source IP address from an RFC 1918 address to a publicly routable address. Firewalls and other devices may reject packets with spoofed addresses. While there are many optional fields in the IP and TCP headers, Craig Rowland argues that the mandatory fields may be more likely to reach the destination intact as they are not stripped off by packet filtering mechanisms or damaged through fragmentation and reassembly [16]. Since covert channels are typically utilized to provide anonymity, ease of detection is an important characteristic. In some cases, the tools can evade detection by a casual observer, such as a naïve network administrator. However, these tools may be discovered by a trained intrusion detection system or by an official dedicated to uncovering covert enemy communications.

Bauer defines important qualities of covert channels when they are used for anonymous communication [4]. In order to increase anonymity, the covert messages must be merged with legitimate messages, either by directly injecting the covert message into the legitimate message or by assuring that there are enough legitimate messages on the network to hide among. The communication between sender and receiver must be "unlinkable." Unlinkable means that an observer can determine that messages are being delivered, but the observer can not tell who communicates with whom. A stronger property, "unobservability," indicates that the observer can not determine if messages are being delivered at all. The term "anonymity sets" refers to the set of all possible subjects who might participate in the transmission or reception of covert messages. A larger the group of possible subjects increases the probability that the participants in the covert channel will remain anonymous. Most strategies defined in this report are linkable; they require that the sender directly communicate with the recipient. Messages may be reflected off a naïve third party to increase anonymity, yet these still may be detected as these linkable channels frequently generate their own network packets for hiding data. Unlinkable systems typically wait to inject their data into packets naturally existing on a network.

4. Current Research and Tools

4.1 The Data Link Layer

Much network steganography development focuses above the data link layer of the OSI reference model. There are reasons for this. Data link layer headers offer limited benefit in wide-area communication. The link layer headers are replaced each time a frame passes through a network layer device. This means that for two systems to communicate covertly using link layer headers, they must be on the same local area network (LAN). Even if devices are on the same LAN, there remains the challenge of anonymity. Most wired LANs switch packets instead of broadcasting them to all connected systems as on a hub. This means that the sender, in order to deliver a message his target, must either send the message directly to the recipient, or intentionally broadcast the message to all hosts on a LAN. While broadcast link layer traffic is not uncommon, it does not aide anonymity. Modification of data link layer headers typically requires low-level control of the network interface hardware. HICCUPS, which stands for Hidden Communication System for Corrupted Networks, is not immune to this challenge.

HICCUPS, by Krzysztof Szczypiorski, takes advantage of the implementation of data link layer headers in wireless LANs [18]. Wireless protocols, such as IEEE 802.11, require that all frames be broadcast over the airwaves since there is no direct physical connection between hosts. While the destination hardware address may be unicast, the signal is sent from the wireless radio in all directions. Additionally, wireless data transmission is an imperfect medium. Interference and noise are common in wireless communications and the protocols allow a certain level of distortion. HICCUPS usurps this flexibility and masks messages in wireless traffic that appears corrupted by interference.

Szczypiorski enumerates three properties of a network environment that make it amenable to coercion. First, the medium must be shared and the recipient must have a high probability of intercepting the message. This is the only essential property for the HICCUPS proposal. Second, an environment with a published algorithm for link layer cipher initialization is advantageous. Third, the environment should provide some mechanism for validating message integrity, such as Cyclic Redundancy Check

© SANS Institute 2006, Author retains full rights.

(CRC). In a network environment possessing these three properties, three hidden data channels can be introduced. The first channel manipulates the initialization vector in the link layer cipher. The second channel relies on forged link layer hardware addresses (MAC addresses). The third channel co-opts the data integrity mechanism. All three properties exist in 802.11 networks with WEP encryption, thus 802.11 networks offer three channels for network steganography. High error rates are common in wireless networks, with CRC error rates commonly reaching 25 percent. This high tolerance for errors presents a greater opportunity for sending secret messages in corrupted headers.

HICCUPS encoded messages can only be deciphered by stations belonging to the "hidden group." Membership in this hidden group is defined during the system initialization phase of HICCUPS operation. During initialization, group members agree upon encoding and decoding schemes and choose whether to protect their transmissions with cryptography. While the frames altered by HICCUPS will be broadcast over radio waves to all listeners, the decoded communication among group members may be unicast, multicast, or broadcast to stations which participate in the hidden group. System initialization also includes incorporation of the traditional confidentially mechanism, cryptography. Enciphering data not only makes it more difficult for the eavesdropper to decode, but it also introduces randomness which makes the subversion more difficult to detect. The procedures for recruitment into the hidden group and the procedures for managing key exchange are left undefined. Szczypiorski suggests using a key exchange algorithm such as Diffie-Hellman, but he warns that the key agreement process may expose the members of the hidden group if it is not carefully implemented.

Encoding and transport of covert messages is implemented in the next two HICCUPS modes. "Basic mode" utilizes hidden data channels in network interface MAC addresses and cipher initialization vectors, as shown in figure 1. The 802.11 standard uses four 48-bit hardware MAC addresses for identifying the source, destination, transmitter, and receiver of wireless frames. 802.11 networks with WEP encryption utilize RSA's RC4 cipher to generate initialization vectors. The initialization vector is a 24 bit field. There are a total of 216 bits per frame available for data hiding in basic mode. Keep in mind that the system is based on purposely injecting seemingly meaningless data into the initialization vector and MAC addresses. Thus, the initialization vector is never actually used for decipherment by a receiving station and the MAC addresses do not have to represent an actual device. The sending station can repeatedly send frame after frame with 216 bits of erroneous data which will only be understood by hidden group members. This is still limited bandwidth in contrast to correctly functioning protocols, thus basic mode is used primarily for exchange of control messages.

Figure 1. HICCUPS Basic Mode

The highest bandwidth is available in HICCUPS "corrupted frame mode." In this mode, the entire data payload can be used for delivery of covert messages. The Frame Check Sequence (FCS) is manipulated to signal to hidden group members that a steganographic message is being delivered inside the data payload (figure 2). Stations that do not belong to the hidden group will discard the corrupted frames assuming that they were damaged in transit. Stations that do belong to the hidden group know the algorithm for creation of bad checksums and how to extract the hidden data. Overall, Szczypiorski estimates that the bandwidth available for steganography is 44 kilobits per second on an 802.11 b network and 216 kilobits per second on an 802.11 g network.

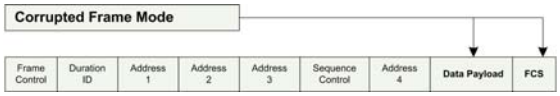


Figure 2. HICCUPS Corrupted Frame Mode

While the HICCUPS proposal offers high bandwidth in contrast to other steganographic systems, it suffers from serious setbacks. As indicated previously, link layer steganography may require low level control over network interfaces. Reception and decoding of steganographic messages can be handled by commercially available wireless cards in monitor mode, a special wireless case of promiscuous mode. However, Szczypiorski could not find any card that allowed manipulation of CRC checksums on outgoing packets via an existing software mechanism. This prohibited Szczypiorski from actually implementing the HICCUPS system.

HICCUPS is not undetectable, either. Wireless intrusion detection systems (IDS) provide one mechanism for detecting intentional corruption of wireless frames [1]. For example, the IDS can track the source and destination hardware address of wireless frames. The purported address can be validated against a published list of known organizationally unique identifiers, the first three octets of a MAC address. Addresses not included in

© SANS Institute 2006, Author retains full rights.

the list can be tagged as anomalous and subject to further investigation. More importantly, wireless IDS typically monitors network performance characteristics such as packet retries and CRC errors. As mentioned previously, a CRC error rate of 25 percent is not uncommon in an area with a large amount of wireless traffic. Rates exceeding 25 percent, however, should trigger an alarm in the intrusion detection system. If there are no interference sources in the same airspace as the wireless system, then frame tampering could be the cause.

4.2 The Network Layer

The network layer is dominated by the Internet Protocol (IP) and its complements ICMP, IGMP, ARP, and RARP. The network layer provides for the transfer of data in the form of packets across a communication network. IP is responsible for routing, the process of selecting a path across a network. It makes wide area networking possible, which means the range of the packets can be global and across disparate network subsystems. Because of this, network layer protocols are popular targets for data hiding.

The Internet Protocol version 4 (IPv4) specification outlines 23 fields to carry routing, quality of service, and fragmentation information. Many of these fields have been hijacked to carry covert communications. For example, the 8-bit Type of Service (ToS) field indicates delay, throughput, reliability, and cost requirements of the IP datagram. Any of these 8 bits can be used to carry a covert payload. Since this field is used infrequently, setting it to any nonzero value significantly increases probability of detection. Hintz suggests using only the delay bit to reduce odds of detection, but this decreases bandwidth to one bit per datagram [9].

The 16-bit IP Identification field is a common target for developers of covert channel tools. The IP ID field is intended to uniquely identify datagram fragments as they reach their destination. According to the Internet Protocol RFC, the field should be "unique for the source, destination pair and protocol for the time the datagram (or any fragment of it) could be alive in the internet" [14].

Craig Rowland demonstrates characteristics of IP ID manipulation with his tool Covert_TCP [16]. This proof-of-concept tool limits its focus to analysis of header manipulation and not optimal data hiding strategies. While the tool is not especially stealthy, it demonstrates the simplicity of usurping the IP ID field for covert message delivery. Covert_TCP simply replaces the 16-bit IP Identification field with a mathematical product of the ASCII value of the character to be encoded. Characters are read from a file and injected one by one into TCP SYN packets. The value injected into the IP ID field is simply the ASCII value of the character multiplied by 256. Figure 3 shows the Wireshark decode of an IP header forged by Covert_TCP. The Identification field is set to 18432, which, when divided by 256 yields 72, the ASCII value for the letter 'H.' The recipient is passively listening for connections and thus never responds to the SYN requests, so a TCP three-way-handshake is never completed. Since establishment of a TCP session is not required, Covert_TCP has a potential bandwidth of 16-bits per packet. However, only 8 bits of the IP ID field are occupied in Rowland's implementation; the rest are set to zero. The method of transforming the ASCII characters into a bit sequence is similar to a substitution cipher, where the encoded value of the ASCII character always produces same encoded value for the field. This significantly increases probability of detection. Rowland suggests that using XOR or actual encipherment will produce a more random result which seems to adhere more closely to the protocol specification.

```
⊟ Internet Protocol, Src: 10.30.21.21 (10.30.21.21), Dst: 10.31.12.85 (1
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 40
    Identification: 0x4800 (18432)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
```

Figure 3. Internet Protocol Header Decode

Two more strategies, proposed by Kamran Ashan, require that the sender and the recipient normally communicate overtly [3]. Supplementary covert information is hidden in their overtly delivered messages. The first proposal takes advantage of redundancy in the Internet Protocol header provided by the Flags field. The Flags field in the IP header contains three bits per packet. The first bit is reserved. The second bit is called the Do Not Fragment (DNF) bit. When set, it indicates to intermediary systems that the packet must not be fragmented. If the third bit is set, it signals that there are more fragments in subsequent packets. The settings of the flags should match the actual context of the packet, yet there is rarely validation. In an unfragmented datagram, all bits should be set to 0.

For Ashan's first proposal to succeed, the sender and the recipient must agree on maximum datagram size, which is known as Maximum Transmission Unit (MTU). Additionally, any intermediary systems must not have a lower MTU. If the size of the packet sent is smaller than the MTU of all systems the packet traverses, then the value of DF is irrelevant. Ashan sets DF to 1 to send a 1, and sets

© SANS Institute 2006, Author retains full rights.

DF to 0 to send a 0. The recipient, knowing the covert significance of the DF bit, stores the values in a buffer and recomposes them into the message. The MTU requirements of this simplistic proposal limit its usefulness in communication across the Internet where paths between hosts may be unpredictable.

Ashan's second proposal exploits concepts from digital image watermarking to scramble the message contents. As with Covert_TCP, the message is delivered in the 16-bit IP Identification field. What separates Ashan's proposal from others is the encoding algorithm. Ashan borrows the concept of "toral automorphism systems" from digital watermarking applications typically used to identify protected images. Toral automorphism systems are mathematical transformations using lattices that deform an image and produce a new one unrelated to the original. It is a complex algorithm for scrambling producing a high level of randomization. Ashan uses the algorithm to develop a look-up table and map each letter to an 8 bit binary value. The selection of the 8-bit mapping is highly random, possibly more random than if they were encrypted. These 8 bits occupy the first half of the IP ID field. The second half is occupied by an independent, randomly generated 8 bits which has no relation to the message. Exploiting the IP ID field allows the sender and recipient to communicate across the Internet. However, both of Ashan's proposals require that the sender and recipient communicate overtly. While this fails Bauer's unlinkability test, the covert payload is characterized by enough randomness to evade some forms of statistical analysis.

The Internet Control Message Protocol (ICMP) is a complement to the Internet Protocol that allows conveyance of error messages and other information at the network layer [15]. ICMP packets are tunneled inside of IP datagrams. The first byte of the datagram is an ICMP type field. There are fifteen types of ICMP messages. The message type determines the format of the remaining portion of the ICMP data. This paper is primarily concerned with ICMP echo and echo reply messages, types 0x8 and 0x0 respectively. These messages are used by the command Packet Internet Groper (ping) to determine whether a host is online and available. ICMP is an integral part of the TCP/IP protocol suite and is always implemented alongside IP. The prevalence of ICMP in IP networks makes it a relevant location to introduce potential covert channels, yet ICMP may be blocked by firewalls because of its abuse during reconnaissance and by Internet worms.

Project Loki, published in Phrack magazine by daemon9 and alhambra, demonstrates the capabilities of ICMP for carrying covert payloads [2]. Loki, the Norse god of deceit and deception, was known for his subversive behavior. The authors of the tool claim that the god Loki masked evil intent in seemingly good and appropriate actions, thus they felt the name appropriate for their clandestine tool. They take advantage of the data field in an ICMP echo message to mask their malevolent payload. The data field in an ICMP echo message is intended to record route information or store timing records to calculate round trip time. The data field by default is typically 24 or 56 bytes long, depending on the host operating system. However, the protocol allows it to be much longer, thus yielding arbitrarily high bandwidth. There is rarely inspection of the data field by host operating systems, firewalls, or intermediary routers, so this field can contain arbitrary data. Loki uses this data field for encapsulating its covert message.

Loki's successor, Loki2, is the only actual implementation of the concept [7]. The author, daemon9, suggests three options for compilation of the tool. The default option compiles an executable that can be run as a visible daemon. The second option allows it to be complied as a loadable kernel module. For maximal stealth, they recommend recompiling the operating system kernel with Loki-enhanced ping. Loki2 utilizes encryption to augment message encoding. Options include weak encoding with XOR, and stronger cryptography with Diffie-Hellmann and Blowfish. Loki2 was developed primarily as a backdoor to tunnel shell commands inside echo requests and replies. To accomplish this, the code allows allocating a pseudo-terminal or piping the commands into a shell.

If the message is correctly encoded in the data field, it may go undetected. There are, however, defenses to counter ICMP covert channels. The simplest is to block ICMP traffic at choke points along the network. However, since this will totally preclude the legitimate troubleshooting uses of ICMP, it should be used with prudence. Firewalls can limit the systems allowed to use ICMP to a trusted group, yet this too can be circumvented. Since the sender of a corrupted ICMP echo only wants to deliver a message and not determine if the recipient is alive, the sender can spoof the source IP address to be from a member of the trusted group. The payload would reach its destination and the owner of the spoofed IP would receive an unsolicited echo reply. The authors of Loki state that the only sure way to disable the ICMP covert channel is to deny all ICMP traffic; this is not totally true. All operating systems have default characteristics for ICMP messages. For example, in Windows XP, the data field is simply the letters of the English alphabet in alphabetical order repeated to achieve the desired packet size. A tcpdump of a Window XP ping is shown in figure 4. Intrusion prevention systems can

© SANS Institute 2006, Author retains full rights.

learn the default payload for common operating systems and deny datagrams with anomalous payloads.

```
Offset   Hex Dump                                          ASCII Dump

0x0000   4500 003c 0b5e 0000 4001 4472 0afe 0108           E..<.^..@.Dr....
0x0010   0a1e 14ce 0000 265c 0400 2b00 6162 6364           ......&\..+.abcd
0x0020   6566 6768 696a 6b6c 6d6e 6f70 7172 7374           efghijklmnopqrst
0x0030   7576 7761 6263 6465 6667 6869                     uvwabcdefghi
```

Figure 4. ICMP_ECHO Decode

4.3 The Transport Layer

Transport layer protocols build on the service provided by IP to support a wide range of applications. Two basic types of service are offered in the Transport Layer. The first service consists of reliable, connection-oriented transfer of a byte stream. This is provided by the Transmission Control Protocol (TCP). The second service consists of best-effort connectionless transfer of individual messages, which is provided by the User Datagram Protocol (UDP). The simple UDP header, designed for rapid delivery of messages, has only four fields for potential covert message injection. Thus, most research focuses on TCP. The twelve fields of the TCP header include many which are rarely inspected and others that exhibit high randomness. For example, a 32-bit TCP sequence number identifies the position of the first byte of the segment in the overall stream of bytes. Sequence number generation requires randomness as a mechanism to mitigate session hijacking.

While Craig Rowland's Covert_TCP exploits IP ID fields by default, it can also hijack the TCP Initial Sequence Number (ISN) for covert payload delivery [16]. During the TCP three-way-handshake, the originator of the connection sends a SYN packet with a randomly derived sequence number. This sequence number is called the Initial Sequence Number (ISN). All subsequent TCP sequence numbers of the same stream follow a predictable increment. As previously stated, Covert_TCP is a simple proof-of-concept tool with little attention to detection avoidance. Thus, the ISN crafted by Covert_TCP is simply the ASCII value of the character to be encoded multiplied by 16777216.

A third method employed by Covert_TCP eliminates direct communication between the sender and recipient by exercising techniques from reconnaissance and port scanning tools such as Nmap. Covert_TCP bounces packets off a remote site which is unaware that it is being used for illicit purposes. This method is called the "TCP Acknowledged Sequence Number Bounce." Covert_TCP forges the source IP address, the source port, the destination IP address, the destination port, and the initial sequence number and the segment. Figure 5 shows how Eve could send a covert message to Bob using the Covert_TCP ASN bounce. The values assigned to the source and destination ports are not important. The source IP address should be spoofed to be that of the intended recipient of the covert message. The destination address should be the IP of a remote server that will remain unaware of its role in message delivery. The same ISN craftwork just described is used here as well.

The packet will be sent from the client's computer and routed to the destination address, the address of the "bounce" server. If the bounce server is listening on the destination port, it will reply with a SYN/ACK. If it is not listening, it will respond with a SYN/RST. The response, however, will be sent to he spoofed source IP, which is the IP of the intended recipient of the covert message. The response will include an Acknowledged Sequence Number (ASN), which is the ISN incremented by 1. The intended recipient, passively listening for ACKs, knows to subtract one from the ASN and divide by 16777216 to retrieve the ASCII character.

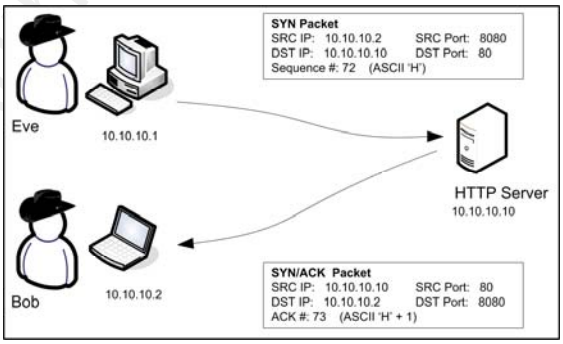


Figure 5. TCP ASN Bounce

There are drawbacks to Rowland's proposals. First, Covert_TCP tampers with values used to provide reliability in TCP, mainly sequence numbers and packet reassembly fields. This tampering makes TCP function much like UDP. Having lost TCP's reliability mechanisms, Covert_TCP slows packet transmission to one packet per second to assure that they arrive in sequence. This delay of one packet per second precludes sending large volumes of covert information. There's an additional problem with the ASN bounce method. Many firewalls and routers include anti-spoofing filters that validate the source and destination addresses of datagrams. If a packet is leaving a protected network and the source IP does not belong to that network, a firewall or router may simply drop the datagram. Finally, using multiplication to encode the message is simplistic and easily detected.

Joanna Rutkowska modifies the model to support true anonymity between sender and receiver with a concept called "passive covert channels" [17].

© SANS Institute 2006, Author retains full rights.

In Rutkowska's proposal, sequence numbers are only modified on traffic normal traffic already created by another user or process on the host. Rutkowska's application, called Nushu, never generates its own traffic. This poses a unique problem: traffic leaving the system already has a defined destination address. Modification of this address would preclude delivery of the overt message to the intended recipient, cause errors in normal traffic on the system, and thus break the channel. In Nushu, the traffic is not sent directly to the intended recipient of the covert message, thus the recipient must have control over some gateway in the path between the sender and recipient of the overt message. Ownership of a firewall, router, or other egress point along the network path would work well.

The Nushu sender is a complex process which must store TCP state information in order to maintain normal traffic flow. To avoid local detection, the passive covert channel is hooked to the Linux kernel's socket handling modules. The primary task of the Nushu covert channel module is to continually modify sequence and acknowledgement numbers from those generated by the operating system kernel to a sequence containing the implanted message. The module is required to modify both sequence and acknowledgement numbers. If only the sequence number were modified on outgoing packets, then the operating system kernel would not understand the acknowledgement number on the corresponding SYN/ACK packet. Similarly, sequence and acknowledgement numbers of all subsequent packets in the TCP session must be modified in order for the operating system kernel to keep up with state information and allow continuation of the byte stream.

As with Covert_TCP, Nushu subverts TCP reliability mechanisms to transport covert messages, however, Nushu redefines the meaning of bits in the sequence and acknowledgement number fields to maintain reliability. Both fields hold a 32-bit value which normally only holds state information. Nushu breaks the field into three bytes for covert data and one byte for control and sequencing. The control byte includes 6 bits for sequencing and 2 bits to mark whether data is stored in the data bytes. Whereas the operating system produced 2^{32} possible sequence numbers, Nushu only has 2^6 values for storing its state information. Empty data bytes are quite frequent since the delivery of covert messages may not match the normal amount of traffic on the system. The limited number of Nushu sequence numbers and the high rate of empty data bytes significantly increases odds of detection. Rutkowska wants to avoid the predictability of the substitution cipher used in Covert_TCP and thus uses a block cipher to encrypt the initial sequence number so that it looks like random data.

The sending Nushu process simply controls TCP sequence information. It must be combined with another process, such as a password sniffer, to generate messages. Messages from the password sniffer can be piped into Nushu. The receiving process does not have to reside in kernel-space. It can be implemented as a simple sniffer capturing all packets that it observes. It can either reside on a trusted gateway or use other mechanisms to force all traffic to its network interface. This means that communication occurs in one direction only, from sender to recipient. This system would work well for capturing information from a computer on a protected network and delivering it to an external host. Adding similar kernel modules to the recipient may allow communication in the opposite direction, thus making the tool capable of delivering commands to the internal, protected host.

Local detection of Nushu sequence number modification may be possible because of the operating system process for socket handling. If a sniffer such as tcpdump is run locally on the compromised host, the sequence and acknowledgement numbers will differ from their value as they pass across the network. This anomalous behavior occurs because the packet capture functions call operating system handles that display packet contents from the operating system perspective, not Nushu's perspective. Rutkowska created an additional module that modifies the way the kernel interprets packets. With this module, a sniffer will call upon modified operating system handles. Thus, a local sniffer will see the same sequence and acknowledgement numbers that are seen traversing the network.

4.4 The Application Layer

The application layer presents limitless opportunities for delivery of covert data. The covert payload can reside either within the protocol headers or be delivered as a payload. Certain requirements must be met before an application layer protocol can provide a useful covert channel. First, use of the protocol should be expected on the system. This helps increase the anonymity set and provide existing traffic to hide in. For example, if the tool hides data in SMTP packets, then it would be advantageous to use an SMTP server as sender or recipient. Otherwise, the covert channel tool may generate SMTP traffic on a system that is monitored and prohibited from sending SMTP, thus triggering an alarm. Similarly, the server may be restricted by a firewall from sending SMTP, thus limiting the group of recipients to those behind the same firewall.

© SANS Institute 2006, Author retains full rights.

The HTTP protocol is a fertile field for embedding covert messages. Even restrictive organizations allow some HTTP traffic to pass from internal hosts to Internet web servers. Inspection of HTTP packets for protocol compliance requires costly software and may break web applications important for business. For example, HTTP can be used as a mechanism for delivering streaming audio and video, chat, and desktop sharing applications. Strict filters that block non-compliant HTTP packets will disable many of these services. Thus, permissive application layer inspection policies allow room for coercion.

Lee Bowyer introduces basic HTTP steganography as a way to deliver covert messages from Trojan horses past stateful inspection firewalls [6]. Trojan horses are frequently utilized by attackers to deliver usernames and password from compromised systems residing behind a firewall. Many Trojans deliver results and accept commands from Internet Relay Chat (IRC) channels since the IRC client is simple to implement. The global availability of the channels and the distribution of the servers make the attacker more difficult to track. However, IRC is typically blocked by many enterprise firewall policies. Bowyer advocates using HTTP for delivery of stolen material. Bowyer suggests attaching the covert message at the end of an HTTP GET request to a web server controlled by the attacker. The fake web server will drop the URL segment of the GET request and parse only the tag, the portion of the message following the question mark. For pulling covert contents inside a firewall protected network, Bowyer requests a normal looking web page with covert messages embedded images. Bowyer states that this is a very difficult hole to plug, as most enterprises need to allow valid-looking HTTP traffic to pass through their firewalls.

A simple proof of concept tool, Reverse WWW Shell, demonstrates how effective HTTP covert channels can be at delivering a payload [20]. In the case of Reverse WWW Shell, the payload is an actual shell executing on a "slave" system. The tool developed by van Hauser shovels the shell through what appears to be a normal HTTP web browser request. The HTTP request, however, is actually a call to Reverse WWW Shell "master" listening for connections from the slave. The person controlling the master can enter commands in the master console and have them execute on the slave. The ability to push a shell through TCP is not new. One of the most functional socket tools, netcat, can accept commands from a shell as input and forward command output to a remote listener over a raw TCP or UDP socket. Reverse WWW Shell, however, uses HTTP headers and tags to mask the shell sent by the slave. The slave call to the server looks like an HTTP GET request to an e-commerce order form with a long CGI tag. In the default configuration, the slave generates HTTP headers similar to a Mozilla 4.0 compliant web browser accepting html and plaintext files in the English language. The CGI tag is actually the base-64 encoded standard output from the shell, as shown in figure 6. The commands entered at the master are sent to the slave masked as an HTTP 200/OK response. Both the master and server components are included in one 260-line Perl script. Everything is customizable, including purported web browser capabilities, choice of shell to bind with, proxy server authentication credentials, and web page to request.

```
GET /cgi-bin/order?M5mAejTgZdgYOdgIO0BqFfVYTgjFLdgxEdb1He7  HTTP/1.0
```

Figure 6. Reverse WWW Shell Slave to Master

While Reverse WWW Shell succeeds in delivering a backdoor over a covert channel, it is not impervious to detection. The slave component must generate its own HTTP traffic in order to deliver the shell to the master. This activity increases the chance of detection. To increase anonymity, the slave should reside on a system known to generate many HTTP requests, such as a public kiosk or the workstation of the one employee who surfs the web all day. Additionally, the slave makes a direct connection to the master. This is easily detected by traffic analysis attacks. As mentioned previously, in some cases the fact that two parties are communicating is more important then what they say. However, delivering an interactive shell seems to necessitate direct communication between slave and master. A passive HTTP covert channel attempting to achieve a high level of anonymity would limit the functionality of the tool. The execution of commands at the master and delivery of the results by the slave may be prone to errors and delay without a direct connection. The author of Reverse WWW Shell, van Hauser, did not explore this possibility in his essay, but the next tool introduces concepts that could make unlinkable communication possible.

Matthias Bauer proposed a protocol called the "Muted Posthorn," which eliminates the communication between sender and recipient of a message [4]. While the Muted Posthorn may not be capable of delivering interactive shells, it is very stealthy in its delivery of messages. In Bauer's protocol, messages are delivered from one web server to another web server via unsuspecting web browsers acting upon standard HTTP mechanisms.

The protocol takes advantage of five HTTP/HTML features: redirects, cookies, referrer headers, elements in HTML code, and active content. An HTTP redirect tells the web browser that the requested document is available at another location.

© SANS Institute 2006, Author retains full rights.

The redirect location can be the URL of a script followed by a list of parameters. In the Muted Posthorn, the URL links to another server complicit in the covert communication and the parameters carry the encoded message. The maximum capacity of this channel is 1024 bytes. HTTP cookies can be used to deliver messages between two servers in the same cookie domain. Two servers are in the same cookie domain if they belong to the same second level domain. For example, if two web servers' domain names end in "icann.org," their top-level-domain is "org," their second level domain is "icann," and then they belong to the same cookie domain, "icann.org." When a web server that wants to send a message, it must wait until it receives a request from a web browser. The server sends the web browser a Set-Cookie command with the cookie domain and a key-value pair up to four kilobytes long. The value can be an encoded covert message. If the web browser then connects to another web server in the same cookie domain, that server can request the key-value pair in the cookie, and thus retrieve the covert message. The client can easily be forced to visit the recipient server with any number of HTTP/HTML mechanisms. Referer headers indicate the URI of the web page that linked to the site in the current request. As with redirects, there is a 1024 byte limit on the capacity of this channel. Another mechanism, HTML elements, causes web browsers to automatically request objects from web servers. For example, the code to include an inline image in a web page and the code to generate page frames can force web browsers to request data from different web servers without user consent. HTTP commands such as redirects can be pushed through HTML elements and escape inspection by web content filters or other observers. For example, the HTML code could include a <META HTTP-EQUIV> tag calling a redirect command deep within the HTML code. Even though the command is not in an HTTP header field, the web browser will act upon the meta-tag and visit the page indicated in the redirect. The final mechanism, active content, includes code executed by the client such as JavaScript, ActiveX, and Macromedia Flash. Like HTML elements, scripts constructed in these active content languages can be used to force web browser redirects, construct invisible forms, and force a client to POST a message without user interaction. Bauer proposes using banner advertisements with hidden frames as a way to distribute the components among many web servers.

The banner advertisements satisfy one of four components of the Muted Posthorn system. These corrupted advertisements are part of the "node maintainer" entity. Node maintainers provide CGI scripts which encode a covert message as contents of a POST request. The encoded message includes both a data payload and headers indicating further processing actions. The possible actions include storing a message in a mailbox, retrieving a message from a mailbox, and forwarding the message to another node. The banner advertisements are linked to from other web pages that are unaware of their role in the covert channel. These web pages are called the "linkers". The linker page simply contains a reference to the URL of the node maintainer. "Senders" and "receivers" represent the entities that knowingly post and retrieve covert messages to the node maintainers. These node maintainers host mailboxes for the senders and retrievers. The messages in the mailboxes are transferred from one node to another via "hapless web surfers." The web surfers visit a web page they trust, but an advertisement on the web page links to a script on the node maintainer, which causes the surfer's browser to download the covert message and post it to another node.

There are two commands in the protocol. The command "To" identifies the node and mailbox that are the ultimate target of the payload. The command "Get" requests messages from a specific mailbox on a certain node. Senders of messages and requesters of mailboxes embed these commands in standard HTTP GET requests that look like unadulterated HTTP requests. Since all of the transactions between senders, receivers, and unsuspecting web surfers are embedded in standard HTTP traffic, it can take advantage of anonymizing web services such as the Onion Router and Anonymizer. These services further enhance the anonymity of the communication. The HTTP headers are typically not modified by firewalls and network address translation has no impact on the message delivery.

The Muted Posthorn system is dependent on a few factors to maintain anonymity. First, the owners of the linker web pages must be willing to link to the node maintainer site. This is one of the advantages of deploying the message transfer scripts in banner advertisements. The linker receives monetary remuneration for linking to the banner advertisements. Additionally, the linker sites must be popular among web surfers. The number of web surfers visiting a linker site directly correlates to the speed a message is delivered from one node to another. A larger number of hapless web surfers also increases the anonymity set in which to hide.

5. Advanced Detection Mechanisms

A number of design flaws were highlighted alongside discussions of the covert channel strategies throughout this paper. Few of the mechanisms for

© SANS Institute 2006, Author retains full rights.

detecting these flaws possessed novel, scientific methodologies. While the specifications indicated that certain header values should exhibit a high level of randomness, the implementations are developed by human programmers using known algorithms and well-documented pseudorandom number generators. It is, in fact, quite possible to predict the values of fields which are intended to be random.

Stephen Murdoch defines a suite of tests used to passively monitor network traffic and identify steganography in the IP ID and TCP ISN fields [12]. The model is described as a "passive warden," which has the capability off observing all packets leaving its network, yet not modifying them as would an "active warden." Murdoch explains that IP ID and TCP ISN fields, while semi-structured, have some unpredictability because of a randomly generated, per-host secret and by use of cryptographic functions. Murdoch details the algorithm for ISN generation in the open source operating systems Linux and OpenBSD. The description outlines hash algorithms, re-keying intervals, and initialization traits. These characteristics lead to well-defined behavior that can be used to identify anomalies potentially caused by steganography. Normal behavior for the IP ID field includes use of a global counter for IP ID that yields sequential values, a sequential per-host IP ID, the fact that the MSB is toggled every 30,000 packets in OpenBSD, and the fact that within a re-keying interval, the OpenBSD IP ID must not repeat. Similar characteristics are outlined for the TCP ISN. Murdoch does not specifically outline the testing methodology or the statistical results of the comparison of original packets and steganographically modified packets. He does, however, claim that with a large enough set of packets, the tests proved that steganography can be detected.

A subsequent paper by Eugene Tumoian describes an elegant analysis platform based on packet traces from tcpdump and neural network models with Matlab [19]. Tumoian's detection method does not require careful study or knowledge of ISN implementation details. A large number of ISNs generated by an unadulterated, trusted operating system stack are collected to form what is known as a "training set." The initial training set is fed into the neural network in order to identify thresholds that may identify ISNs not generated by the trusted stack. If actual ISNs fall within the boundaries of the threshold, the neural network will continue learning from observed values. If the ISNs significantly exceed the threshold, the ISN is considered steganographically altered. In Tumoian's experiment, 1500 actual ISNs each are read from multiple operating systems, including Linux and Windows distributions. These are loaded into Matlab using a neural network training script. For monitoring of actual traffic, WinDump log files are read one-by-one by the Matlab script. The results show a high level of precision in detecting the presence of covert channels and a low occurrence of false-positives. Without a-priori knowledge of the sender's operating system, Tumoian's neural network can detect covert channels.

6. Conclusion and Future Work

Many of the concepts outlined in this report discuss the simplicity of altering published protocols for an illicit purpose. Two primary motivations were discussed: delivering a secret message and sending shell commands over a covert backdoor. The underlying intent of these actions may be to evade government scrutiny, to commit illegal acts, or perhaps to perpetrate some benevolent purpose. New specifications are frequently published by the Internet Engineering Task Force (IETF) and the Institute of Electrical and Electronics Engineers (IEEE). It seems that little has changed since Steven Bellovin raised awareness of flaws within protocol in 1989. While standards organizations frequently incorporate mechanisms for protecting the confidentiality, integrity, and authenticity of network transmissions, opportunities for subversion of the protocols increase. Some measures can be taken during implementation of the protocols to assure that they are not used for unintended purposes. For example, for header fields with a limited number of defined values, the implementation should validate that no unintended values are sent. The same validation can be done at various points along the network path, such as routers, firewalls, and intrusion analysis systems.

If preventative mechanisms are not developed, new opportunities exist which increase the range of possibility for protocol subversion. For example, Internet Protocol version 6 is implemented in almost all current operating systems and routers. However, it is not widely used nor understood by network administrators. For this reason, some intrusion detection systems consider the mere presence of IPv6 as a potential indication of a compromise [10]. Another protocol which promises widespread adoption is IEEE 802.16, also known as WiMax. WiMax seems to pledge wireless functionality similar to that found in 802.11 networks. With WiMax, strategies such as those used in HICCUPS may deliver covert channels that are broadcast over many square miles instead of just a few hundred square feet. This would make physical location of the communicating devices seemingly impossible.

© SANS Institute 2006, Author retains full rights.

References:

[1] AirDefense. Wireless LAN Security: What Hackers Know that You Don't. Online: http://www.airdefense.net/

[2] Alhambra. "Project Loki: ICMP Tunneling." Phrack Magazine, Volume 7, Issue 49, August 1996, at http://www.phrack.org/phrack/49/P49-06.

[3] Ashan, K. and D. Kundur. Practical data hiding in TCP/IP. Proceedings ACM Workshop on Multimedia Security, 2002.

[4] Bauer, M. New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets. In Samarati P, Syverson P, editors. Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, 2003 Oct 30; Washington, DC. ACM Press. p 72-78.

[5] Bellovin, S. 1989. Security Problems in the TCP/IP Protocol Suite. ACM Computer Communications Review, 19(2), March 1989.

[6] Bowyer, L. Firewall Bypass via protocol Steganography. Online posting. 2002 Sep 22. http://www.networkpenetration.com/protocol_steg.html.

[7] Daemon9. LOKI2 implementation. Phrack Magazine, 7, September 1997. Online posting: http://www.phrack.org/show.php?p=51&a=6.

[8] EFF: Electronic Frontier Foundation. Privacy – Crypto – Key Escrow 1993-4 (US): Clipper / EES / Capstone / Tessera / Skipjack Archive. March 13, 2003. Online Posting: http://www.eff.org/Privacy/Key_escrow/Clipper/.

[9] Hintz, A. Covert channels in TCP and IP headers. Presentation at DEFCON 10. August 2-4, 2002. http://www.defcon.org/images/defcon-10/dc-10-presentations/dc10-hintz-covert.ppt.

[10] Internet Security Systems. Signature Database: Native usage of the IPv6 protocol has been detected on the network. June 13, 2003. Online: http://xforce.iss.net/xforce/xfdb/12275.

[11] Koops, B. Summary of International Crypto Controls. January 2006. Online posting: http://rechten.uvt.nl/koops/cryptolaw/cls-sum.htm.

[12] Murdoch, S. Embedding Covert Channels into TCP/IP. 7th Information Hiding Workshop. Barcelona. June 2005. pp 247-261.

[13] National Institute of Standards and Technology. Trusted Computer System Evaluation Criteria. August 1983.

[14] Postel, J. RFC 791 - Internet Protocol.

[15] Postel, J. RFC 792 - Internet Control Message Protocol.

[16] Rowland, C.H.: Covert channels in the TCP/IP protocol suite. First Monday 2 (1997) http://www.firstmonday.org/issues/issue2_5/rowland/.

[17] Rutkowska, J. The implementation of passive covert channels in the Linux kernel. 21st Chaos Communication Congress. December 2004. Berlin. http://www.ccc.de/congress/2004/fahrplan/event/176.en.html.

[18] Szczypiorski, K. 2003. HICCUPS: Hidden Communication System for Corrupted Networks. The Tenth International Multi-Conference on Advanced Computer Systems, 2003 Oct 22-24; Międzyzdroje, Poland. pp 31-40.

[19] Tumoian, E. Network Based Detection of Passive Covert Channels in TCP/IP. IEEE Conference on Local Computer Networks. 2005. pp802-809.

[20] van Hauser. Placing Backdoors through Firewalls. Online Posting. May 3, 1999. http://www.thc.org/papers/fw-backd.htm

© SANS Institute 2006, Author retains full rights.

Last Updated: September 4th, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
Security Leadership Summit & Training	Dallas, TXUS	Dec 03, 2015 - Dec 10, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced