INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

New Approach towards Covert Communication using TCP-SQN Reference Model

Dhananjay M. Dakhane
Department of Computer science & Engineering
Sipna College of Engineering & Technology, SGBAU, Amravati, India
Dr. Prashant R. Deshmukh
Department of Computer Science & Engineering, Dr. P. D. Polytechnic Amravati, India

Abstract:
Covert channel stands for transfer of unintended information. It allows the attacker to send as well as receive the secrete message without being identified or detected by the Network administrator or the warden in the network. There are several ways to implement such covert channels; one of them is storage covert channel where data is sent through certain header field of TCP, IP protocol stack. However there is always some possibility of these covert channels being identified. Here, we propose a new covert channel technique, 'TCP-SQN Reference Model'. In this technique a new covert channel is created in Linux kernel, using TCP Sequence Number as a reference for sending the covert information. The idea of our proposed model is, sender is not actually embedding the secrete message into the TCP-SQN filed; instead the sender uses it as a reference, to convey the secret message to the receiver. As sender is not actually modifying the TCP-SQN filed, the sequence number is observed as a normal packet distribution, which is created by any Linux or BSD Kernel. So it is difficult to distinguish overt and covert packet in the network.

Keywords*: Covert channel, TCP/IP, TCP Headers, TCP ISN, TCP-SQN (Sequence Number)*

1. Introduction

Security of network is one of the most complex and difficult problem for the most of the organizations and governments. Use of the internet, actually increase the complexity problem of network and information security. There are many alternatives such as Firewall, IDS, VPN etc. These alternatives may restrict the known attacks from the external attackers. The real threat is from insider attackers. The vast number of insiders is largely trusted in order to maintain productivity of organization. Insider attackers may use the vulnerability of the protocol to create hidden channels or covert channels. The concept of Covert Channel is defined in [1][2]; it is a communication channel that allows a process to transfer information in a manner that, it violates the security policy. This definition includes covert channel in a single system. In the recent years the focus is shifted to the network based covert channel [3]. If header field of a network protocol is used for the covert communication, huge amount of information can be covertly transferred. The number of different protocols used in the internet, makes it ideal as a high-bandwidth vehicle for covert communication. The capacity of covert channel in computer networks has greatly increased because of new high-speed network technologies and this trend is likely to continue. Even if only one bit per packet can be covertly transmitted, a large Internet site could lose 26 GB of data annually [4].

In this paper, we propose a new model for creating a covert channel of TCP/IP stack as a 'TCP-SQN Reference Model' for new way of covert communication. This proposed model, consider that the sender is not actually embedding the secret data into the TCP ISN filed, instead, the sender will only use it for the reference to convey the secret message to the receiver. As sender is not modifying the semantics of the TCP SQN filed, it is difficult to distinguish between the overt and covert packet in the network.

2. Related Work

Rowland initially discovered covert channels in the TCP Initial Sequence Number (ISN) field [2]. The ISN is only transferred when a new connection is established and has a size of 4 bytes.

Rowland also developed an enhanced version of the ISN-based covert channel with the goal to hide the sender's address. Therefore, a bounce server is introduced. The bounce server is used by the sender to send messages to the receiver. Therefore,

the senders send a spoofed TCP packet to the bounce server. The packet contains the receiver's source address and thus, lets the bounce server respond to the receiver that receives a packet which does not contain the sender's address. The bounce server will increment the ISN that is transferred as acknowledgement number to the receiver that has to decrement the acknowledgement number to get the original ISN value. Rutkowska developed an enhanced ISN-based passive covert channel by indirectly initiating a TCP connection to transfer hidden information [4]. Instead, a covert channel sender waits for a regular TCP connection and modifies an ISN generated track by an ISN modification layer in the Linux kernel [4].

Another approach is to use TCP and UDP port numbers to send hidden messages [6]. J Giffin used TCP timestamps to embed covert payload [7] (timestamps are an optional header component of the protocol). This author applied a minimal delay to create a covert timing channel in this way.

In storage based covert channels certain fields of header in the packet is used throughout the stream. All of the packets of the corresponding stream contain covert data; this is the key feature of the storage based covert channels. Previous work done by Joanna Rutkowska and Steven J Murdoch gave the two schemes for embedding the covert channels in TCP/IP. Joanna Rutkowska designed the scheme called "NUSHU"[4], and Steven Murdoch designed the scheme for covert channels called "Lathra"[5].

"NUSHU" encrypts the data before actually embedding it into TCP ISN field [4]. This results in normal distribution unlike that is generated by Linux and so will be detected by other TCP tests. "NUSHU" also exhibits characteristics of its own which may be exploited. The encryption operated by DES encrypting IV (Source port + Destination port + Source IP address + Destination IP address) with a shared key, then XORing the first 32 bits of the resulting key stream with the hidden data. When IV collisions occur, the ISNs can be XORed to remove the key stream; the result is XOR of two plaintexts.

If these plaintexts are the same, as in the case when the data is not being sent, the result would be zero. In other cases redundancy in encoding would be apparent [4]. So in certain cases this protocol fails. While "Lathra" [5] designed by Steven j Murdoch and Stephen Lewis works perfectly in those cases.

In both of the above covert embedding schemes headers of the TCP/IP layer are modified, thus leaves some scope for warden in the network to detect these channels. To be totally undetectable while embedding the covert data in the packet we should not modify anything in the headers whether it is TCP header or IP header. In such a way it becomes difficult for warden to distinguish between overt data packet and covert data packet.

3. Threat Model

We assume that Alice is a part of secure network environment where a warden is present. Alice wants to covertly communicate with Bob who is operating from outside of the network. Bob will receive the covert data which is sent by the Alice where warden is continuously monitoring the network traffic. The idea of our threat model is shown in figure no. 1. We assume that the gateway system is monitoring all the traffic along with the statistical based analysis of the network traffic. In the figure the green region is consideration of total traffic bandwidth of overt communication and the red region represents the network traffic bandwidth consume by the process exploiting covert channel. This remains very insignificant difference in covert channel in corresponds with overt channel, making it difficult for the gateway system in order to detect the traffic bandwidth consume by the covert channel making it undetectable.

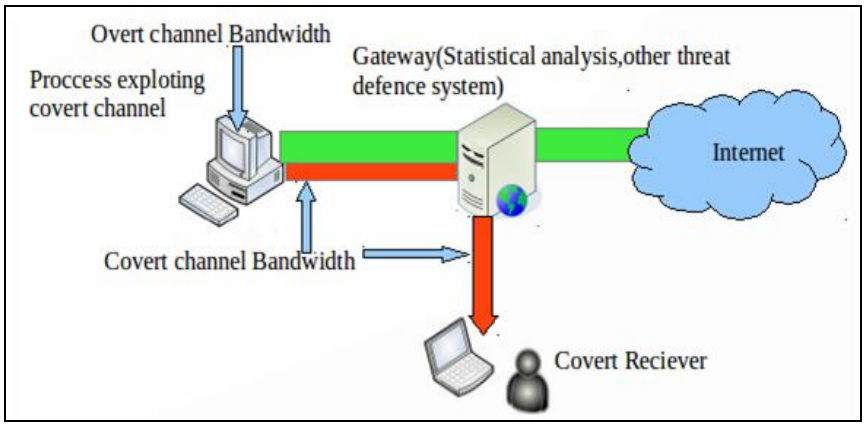

Figure 1: Covert Channel System

In our model, Alice having the application programs to insert the covert data into the live network packet. Similarly Bob also have equivalent application for extracting the covert data from those packets. The data which Alice is sending to the Bob is potentially sensitive data of the organization. The main task of the warden in the network is to detect and stop such transmission of the sensitive data. To perform such communication successfully without being detected, Alice and Bob operate over the same shared secret and symbols.

Our model is designed for high security environments where network is not a free channel, but is instead frequently monitored or restricted against unauthorized usage. The packet loss, duplication and reordering of the packet is not permitted.

4. Features of TCP SQN Reference Model

- NORMAL DISTRIBUTION OF ISN
  In this model packets are generated by the kernel itself. Therefore when we use Linux kernel 3.0 for the implementation of this model; the semantics of TCP ISN generation specified in the RFC1948 are followed. So the TCP SQN used by the packets containing covert data, cover the same space as the overt data packets. That's why the normal SQN distribution for covert data packets is observed.
- PERSISTENT CONNECTION
  Previous approaches by Rowland, Rutkowska and S. Murdoch uses new connection for each unit of covert data. While this model uses a single persistent connection for communicating the whole data so bandwidth of this covert scheme is extremely high compared to "NUSHU" [4] and "Lathra" [5].
- RELIABLE COVERT TRANSMISSION
  In the case of packet loss, Kernel itself performs packet retransmission, as in our proposed model all of the tasks of packet transmission are done by the kernel itself. This enhances the reliability of the covert channel for delivery of the covert data.

5. Proposed TCP SQN Reference Model

5.1. Proposed Model

When two parties need to transfer the data using TCP, the sender machine will create a new TCP connection. As sender machine initiate the connection, it generate the first sequence numbers i.e. TCP ISN. The sequence number has dual role. If SYN flag is set, then it is initial sequence number (ISN). When SYN flag is clear, then the sequence number is the accumulated sequence number of the first data segment of the current session. The TCP ISN must be chosen such that the sequence numbers of new incarnations of a TCP connection do not overlap with the sequence numbers of earlier incarnations of a TCP connection [5]. Storage covert channels using TCP SQN field as a reference, to be undetectable it is necessary that ISN's generated by these channels should cover the same space as the ones generated by the system without modification. Previous research shows that whenever the semantics of the packet are disturbed, it becomes detectable by the warden present in the network. As we are using TCP SQN field as the storage channel for our covert data; it is most important that ISN numbers generated by our protocol should look like normal ISN distribution. To achieve this we use ISN generated by the Operating system's kernel itself and do not generate any random numbers for ISNs. This will ensure us that whatever ISN number we are using, will look like any other ISN generated by the same system. In this way we will automatically follow all the specified structures and semantics used by the kernel. Now to convey the covert data we use the TCP payload; this will contain the key, using this key we can extract the covert data from kernel generated TCP SQN number. Basically this key is the sequence of our covert data bits in the TCP SQN filed. This key can be distinguished throughout the payload and each byte of the key are placed at different positions in the TCP payload. We call these positions as a data pointer. These data pointers contain the symbols from the symbol-table, which is usually used in the covert communication. This symbol table contains unique symbols for positions in the sequence.
So actually sender will append the TCP payload and not the header in order to send covert data. Hence without disturbing any header or packet semantics covert data can be embedded into the packet.

5.2. TCP SQN Reference Model Implementation

In our propose TCP SQN Reference Model the loadable kernel module (LKM) will embed the covert data into the TCP payload. However the covert data in the TCP payload would not be the actual covert bytes that we want to send. The reference positions which indirectly pointing to individual bits of kernel generated TCP sequence number will be in the TCP payload. Figure no. 2 shows the TCP SQN Reference model architecture. To allow us to determine whether the successful covert communication will take place using SQN Referencing model; we have implemented sample proof-of-concept test. We created client and server that would communicate over TCP. These applications are written in java and tested on Ubuntu 12.10. With the help of APIs provided in java we used the sockets of the system.

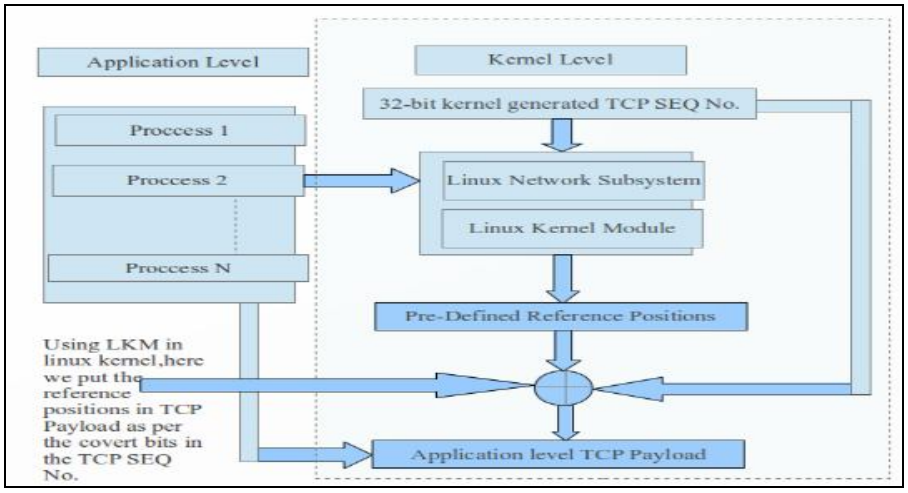

Figure 2: TCP SQN Reference Model

As java uses systems own naive API to perform operations; the packet generation and transmission is done by the system i.e. kernel itself. Therefore retransmission of the lost packet is already ensured.

We wrote two kernel modules that used packet_mangle and netfilter libraries for sender and receiver respectively to inject and extract the data into and from the packet. By using packet capturing tool Wireshark [9], we could confirm whether the interviewing network would support the covert channel. In our first test, we implemented the approach using two computers running Ubuntu Linux 12.10 with a direct network link. Monitoring at the client showed that the packet correctly arrived with its covert payload, allowing the client to harvest this information. We then repeated the experiment using a DLink Ethernet switch between the two hosts. The switch did not affect the transmission, and the covert data is received correctly. In our third test we connected the same two computers with the mobile data cards with route-able IP addresses. Now the two computers are practically in two different networks. As expected the result of this test, the covert data received on the destination correctly allowing communicating information covertly.

In all of the above tests the data packets reached to the destination correctly; also no anomaly is observed in the packets when monitored using wire-shark which allowed successful covert communication in respective networks using SQN Reference model.

5.3. Evaluation of TCP SQN Reference Model

This implementation allowed us to create a covert channel simply because all the conventions and the semantics of packet generation are followed by the kernel and just before leaving the machine packet mangle libraries allow us to change the payload. So it is not checked at all whether the packet was altered before it reaches to the destination.

Here in this covert channel all of the data is transferred through a single persistent connection so all of the data bits use the same connection; while in conventional covert channels the bandwidth is reduced to few bits per connection only. So we can say that ISN reference model has higher bandwidth as compared to conventional covert channels like NUSHU [4] and Lathra [5].

6. Conclusion

In our propose model we are using all ISNs generated by the system itself so it is impossible to differentiate between ISN number with covert data and ISN without covert Data. Pierre Allix [8] also states that correct implementation of this can create totally undetectable channel, however the bandwidth of such channels is very low; only few bits per connection. In our case we are using the single persistent connection for the whole session and all of the communication takes place through this single connection so sour covert channel is having larger bandwidth than conventional covert channels.

7. References

1. US Dodd, (1985). Trusted Computer System Evaluation Criteria. http://csrc.nist.gov/publications/history/dod85.pdf
2. Rowland, Craig (1996). Covert Channels in the TCP/IP Protocol Suite. http://www.firstmonday.org/issues/issue2 5/rowland/
3. S. Zander, G. Armitage, P. Branch (2007). A Survey of Covert Channels and Counter measures in Computer Network Protocols. (Accepted for publication in IEEE Communications Surveys and Tutorials).
4. Joanna Rutkowska, (January 2004). The implementation of passive covert channels in the Linux kernel. Speech held at the 21st Chaos\Communication Congress, Berlin and Germany. http://events.ccc.de/congress/2004/fahrplan/les/319passive-covert-channels-slides.pdf.
5. S. J. Murdoch, S. Lewis. (June 2005). Embedding Covert Channels into TCP/IP. In Proceedings of 7th Information Hiding Workshop.

6. T. Borland (January 2013). Guide to encrypted dynamic covert channels. http://turboborland.blogspot.com/2008/12/guide-to-encrypted-dynamic-covert.html.
7. J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts (2003). Covert messaging through TCP time stamps. In Proc. 2nd International Conference on Privacy Enhancing Technologies, pages 194{208}.
8. Pierre Allix, (2007). Covert channels analysis in TCP/IP networks.
9. The Wireshark Foundation (2013). "Wireshark," http://www.wireshark.org