A Performance Analysis of Authentication Using Covert Timing Channels

Reed Newman, Raheem Beyah

Communications Assurance and Performance Group, Computer Science Department,
Georgia State University
Atlanta, GA, USA
jnewamn8@student.gsu.edu, rbeyah@cs.gsu.edu

Abstract. Authentication over a network is an important and difficult problem. Accurately determining the authenticity of a node or user is critical in maintaining the security of a network. Our proposed technique covertly embeds a watermark, or identifying tag, within a data stream. By implementing this model on a LAN and WLAN we show that this method is easily adaptable to a variety of networking technologies, and easily scalable. While our technique increases the time required for data to be transferred, we show that the throughput of the link during the brief authentication window is decreased by no more than 8% in a switched LAN and 11% in a WLAN. During our empirical analysis we were able to detect the watermark with 100% accuracy in both a LAN and WLAN environment.

1 Introduction

With the large amount of network applications in existence today, it is becoming increasingly difficult for network administrators to police the traffic on their networks. Additionally, these network applications are increasingly carrying sensitive information; as the sensitivity of the information traversing the network increases, so too must the level of security within the network.

A key step in securing a network is the authentication of all nodes[1] on that network. Authentication of nodes allow perimeter devices the ability to determine if a node's requests should be granted. Reliable node authentication helps to ensure the basic elements of network security: positively identifying a node, allowing that node specific access privileges, and holding that node accountable should it compromise the security or productivity of the network.

Leading industry approaches, such as Cisco's AAA Server and Cisco's NAC model [1], rely on user name/password, S/Key, Token Cards or system profiling to authenticate users. While all of these methods provide a reasonable level of security, each are expensive and have been shown to have flaws [2,3]. Additionally, simple

1 Authentication relating to nodes can easily be extended to users. When referencing nodes, unless otherwise noted, the statement applies to nodes and users.

access control lists have proven to be ineffective as spoofing IP and MAC addresses is a trivial task, allowing attackers to easily gain access to systems and networks.

By not requiring strict node authentication, joining a network is a trivial matter. An attacker may obtain a physical connection to a network, enter the network through a wireless access point, circumvent 802.1x via phishing or other known exploits, or access the network via a VPN connection. By requiring each node joining the network to authenticate itself, the overall security of the network is increased making systems within the network harder to breach.

We propose to embed a watermark, or signature specific to a node, within a data flow using inter-packet delay. The watermark will be embedded between select packets in a covert manner, such that it is difficult to detect and does not decrease the throughput of the link significantly. Our model can be used to supplement the above methods or be used as a standalone method of authentication. Our technique does not require synchronization between the nodes' clocks, and can easily be adapted for user authentication.

We acknowledge that our proposed method, a method advocating security by obscurity, is not an impervious solution but assert that the additional layer of security increases the difficulty for an attacker. We believe that providing authentication in a covert manner adds the same level of benefit one achieves when changing an internal server's (sftp, ssh, etc.) port from a well-known port to a random port. The attacker would require not only a technique by which to break into the server (e.g., a buffer overflow attack, etc.), but must also find the server, via port scanning, over a range of 64,000 ports without being detected. Further, this work evaluates performance of using timing channels in general to transmit data. Our specific application of this method uses this channel to communicate authentication information.

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 details the covert timing channel model, Section 4 outlines the experimental setup and procedure, Section 5 provides analysis on the model, Section 6 provides experimental data, and Section 7 contains our conclusions and future work.

2 Related Work

The idea of using inter-packet delay (IPD) for node identification is not necessarily new. There have been several different applications regarding the use of IPD in recent academic research. The majority of the work focuses on the detection of stepping stone connections (i.e., intermediate systems attackers use to launch attacks and insulate themselves from detection), with most solely considering interactive (SSH) traffic.

The authors of [4] extend the work done in [5,6] to detect a correlation between stepping stones. In [4], the authors use a binning technique to partition the data stream allowing the encoded watermark a greater level of robustness against timing perturbations and repacketization within the stepping stone links. In [7], the authors extend the work of [8] in an effort to defeat chaff packets (e.g., structurally correct packets inserted within the data flow to obfuscate a pattern) that may be inserted by a node within the stepping stone chain. This is done by decoding

watermarks from all possible subsequences of a downstream flow, and choosing the "best" watermark (defined as the watermark with the least hamming distance from the original watermark).

Peng, et al. [9] investigate the secrecy of active watermarking. The authors develop an attack technique that infers important parameters of the watermarks, and also recovers and duplicates watermarks through the stepping stones. The authors of [10] attempt to detect stepping stone connection correlations of SSH traffic by creating a logical partition between ON and OFF periods of usage. By examining the IPD of connections and determining if the OFF period transitions to an ON period within a certain threshold, it can be reasonably stated that these connections are related. It is interesting to note that this is an entirely passive technique.

In [11], provable upper bounds are set on the number of packets required to confidently detect encrypted stepping stone streams with proven guarantees of a low false positive rate. The methods in [11] also take into consideration the usage of chaff packets, and provide bounds on the amount of chaff needed by the attacker to evade detection. The model proposed by [12] is similar to that of [11], with the addition of wavelets. In [12], the authors attempt to differentiate between the short term behavior of the stepping stone streams, where timing perturbations and chaff packets can mask the correlation between connections, and the long term behavior of stepping stone streams.

Wang and Reeves [5] propose a watermark-based scheme that can detect correlation between streams of encrypted traffic. However, the assumption is made that the attackers timing perturbations are independent and evenly distributed. Should the traffic be disturbed in another fashion, such as with the insertion of chaff packets, their method will show a decrease in accuracy. Zhang, et al. [13] provide an upper bound on the number of packets required to detect attackers in stepping stone networks when chaff packets and timing perturbations exist simultaneously.

The authors of [14] show that VoIP encoding scheme can easily contain a covert channel by altering the least significant bit. They provide analysis on the bandwidth and the amount of data transferred. Wang, et al. [6] show that encrypted VoIP calls hidden through an anonymizing network can still be traced, using a technique similar to that of Paxson [10].

Only tangentially related to our work, the authors of [15] studied the loss of anonymity in a flow-based wireless mix network under flow marking attacks, in which an adversary embeds a pattern of marks into wireless traffic flows by electromagnetic interference. They asserted that traditional mix technologies are not effective in defeating flow marking attacks in wireless networks. They proposed a new countermeasure based on digital filtering technology. The authors of [16] introduce a covert timing channel model based on the presence or absence of packets arriving during a specific time period. The authors attempt to detect this covert timing channel over IP using a similarity comparison between packet inter-arrival times.

The concept of covert channels was first introduced by B. Lampson in 1973 [17]. Covert channels at the network and transport layers in TCP/IP were first investigated by Rowland [18] and Fisk, et al. [19]. Currently software such as Covert_TCP [19] and Nushu [20] are available to hide data within TCP headers. Project Loki [21] has also shown that ICMP packets are capable of carrying covert

information within their headers. By moving the covert channel to the application layer, detection of covert channels becomes even more difficult. It was discussed in RFC 3205 [22] that HTTP be used as a carrier for other protocols; this was obviously meant for the covert encapsulation of the protocols. Several tools are also available to tunnel protocols through HTTP, like Lars Brinkhoff's httptunnel [23], primarily for the purpose of evading firewalls.

The predominate focus of these works, with the exception of [6,14-19,21-22], are on detection of stepping stones using SSH style traffic. SSH traffic can be classified as high-latency traffic, caused by user pauses in commands being issued. In [14], data in the least significant bit of the VoIP stream is altered to create a covert channel. Wang et al. [6] use a technique similar to that of Paxson [10] to correlate between VoIP flows hidden by an anonymizing network. Additionally, all of the works discussed above, with the exception of [15-19,21-22], focus on wired networks. In [17-18,21-22], the covert channel is actually embedded or encapsulated within another protocol.

We propose a general model (protocol independent) for node authentication that is able to take advantage of low-latency traffic and be effective on both wired and wireless networks. We evaluate this model experimentally and address the performance and accuracy of our model. This model does not require a large amount of overhead **during the brief authentication window**, using no more than 8% of the available bandwidth during testing in a switched LAN and no more than 11% in a WLAN.

3 Covert Timing Channel Model

Through the application of Steganography to a network flow, we achieve *Covert Timing Channels*, which are defined as parasitic communication channels that draw bandwidth from other channels, via the disruption of event timing relative to other events, in order to transmit information [24-25]. Traditionally *Covert Timing Channels* are used for the transmission of messages, and likewise, to provide authentication, we transfer a watermark or signature specific to a node, through this channel.

Particularly, we disrupt the inter-packet timing within a data flow over a network (specifically using the TCP and UDP protocols). By adding minimal amounts of delay to select packets within the flow, we add a watermark to the flow in such a way that it is unique to a specific node. By delaying packets within a data stream, even minimally, we detract from the bandwidth of the network and thus increase the time required for a given task. We will show that bandwidth degrades proportional to the amount of delay needed to accurately transfer a watermark, the length of the watermark, and the rate at which the node must be re-authenticated.

Disrupting the inter-packet timing within a data flow is inherently volatile, especially when considering outside influences such as the overhead incurred from the use of TCP. Given occasionally network volatility we must assume that sending one watermark may not provide adequate authentication. Additionally to provide a greater degree of robustness, in certain cases it may be necessary to increase the

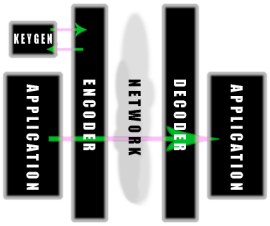
frequency of watermark transmissions. For instance, a university system may not require the node to re-authenticate itself often whereas a military institution, which places a higher priority on security, may. Resending the watermark increases the security provided by the model, allowing the host network to ensure that the node it authenticated originally is still that node. In Section 4 we show that the increase per additional watermark is small, resulting in a minimal decrease of network throughput. Additionally, we provide analysis on the amount of delay required to accurately detect the watermark, versus the percentage error of false negatives.

The degree of natural delay varies from one network type to another. For instance, assuming that network delay is the only factor, the time required to transfer a file over the Internet or VPN connection is greater than the time required to transfer that same file over a wireless network (from one local computer to another); the time required for the transfer of that file over the wireless network is greater than the time required to transfer the file over a 100Mbps LAN connection. With the addition of network congestion, packet loss, etc., the distinction between network types becomes even clearer. As such, the addition of delay required by our model dynamically varies depending on what type of network the user is using. Our model calculates the minimal delay required to accurately embed a watermark into the data flow by using the round-trip time (RTT) of the first ten data packets received per connection over a TCP connection. This approach will not work over a UDP flow, as there are no acknowledgment packets being returned to the sender. If UDP is being used, the client will ping the server several times and calculate the delay based on the RTT provided.

An overview of our model, graphically represented in Fig. 1, is as follows. Initially a bootstrap phase is entered. During this phase, a kernel module, called the *Encoder*, is loaded and an initial watermark is created. The *Encoder* associates itself with an application at the transport layer and acts as a filter allowing for the queuing of packets, so that delay may be added. An additional kernel hook, called the *Detector*, registers itself to determine the RTT of the first ten data packets sent, which will determine the amount of delay added to the data flow. The initial watermark is currently generated from a password, but can be altered to be generated from another input; for example, using a serial number unique to a node (or other identifying information) in addition to a password would allow for the authentication of the user/machine pair. To generate a watermark, our password is hashed using the SHA-1 hashing algorithm, translated from hex to binary, and truncated to the desired length of the watermark.

Two delay values are used within the watermark. A high value is used to represent a binary 1, and a low value used to represent a binary 0; the true high (λ) and low (ω) values are set by the *Detector* with $\lambda > \omega$ and ω being greater than the average delay between packets. When delaying a packet that is part of the watermarked sequence, the binary watermark is consulted to determine if the delay to be added is λ or ω; if the binary value of the delay is a 0, the value represented by ω is used, with the value of λ being used if the binary value of the delay is a 1.

As data is sent from the application, it is registered with the *Encoder*. The *Encoder* tracks the number of packets being sent, as well as the start time of the data flow. From a predetermined level of security configured by the user, the *Encoder* will embed the watermark within the data stream after every β number of packets, or after



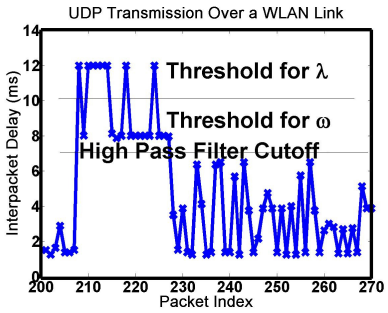


Fig. 1. **Fig. 2.** **Fig. 3.**

Fig 1. A work flow depicting the Covert Timing Channel Model. Fig 2. Depiction of the Experimental Testbed. Fig. 3 Watermarks in UDP traffic transmitted over a WLAN link. Here λ = 12 ms and ω = 8 ms, with the Watermark = {1 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0}.

γ units of time have elapsed. Additionally, the user may configure the watermark to be rehashed using the current watermark as the key to create a new watermark after δ packets have been transferred, or to create a new watermark after ζ units of time have elapsed.

From this point, the data is transferred over the network to the receiver where another module, a kernel hook, decodes the signal; it is not required that a kernel hook be used in this instance, the application itself could provide this service. Abstractly, we shall refer to this entity as the *Decoder*. Prior to decoding the watermark, the data flows' inter-packet timing sequence is subjected to a high-pass filter which filters out the majority of the un-encoded network traffic. Two methods of decoding the watermarks were investigated. These methods are as follows:

1. The Simple Threshold Method – A cutoff point is determined by taking the mean of the IPD not set to zero by the high-pass filter. If a packet has been delayed by an amount greater than the cutoff, it is considered to represent a 1, otherwise it is considered to represent a 0.
2. The Multi-Threshold Method – Using a moving window of two, the values are multiplied together and stored in a separate array. Doing this places the number pairs into three distinctly separate categories, 00, 01/10, and 11. It is difficult to determine the order of the pairing in the 01/10 category; as a solution, the original stream is reviewed, and the greater value is determined to be the 1 value.

Due to space limitations, only the Simple Threshold Method was used during testing.

Should the flow be found to not contain the watermark after a certain threshold, as determined by the level of security required, the *Decoder* can be configured to signal a firewall, or other external device, in an effort to alert a user or disallow continued traffic from the source. There are many additional steps that could be taken, each specific to an individual organization's needs, and beyond the scope of this work.

4 Experimental Setup and Procedure

To test the Covert Timing Channel Model in a controlled environment, an experimental testbed was constructed. The testbed was comprised of a Lenovo 3000 C100 laptop running Fedora Core 4, kernel ver. 2.6.16-1.2111_FC4 with 512 MB RAM as the client sending the watermark. One server, called Server 1, used in the testbed was a custom build desktop computer running Fedora Core 6, kernel ver. 2.6.20 with a 3.0 GHz processor and 1 GB RAM connected on a local network with the client by both a wired and wireless connection. An Airlink101 wireless card was used in the Lenovo client, and a D-Link wireless USB adapter was used in Server 1 for the wireless testing. A Netgear 802.11b router was used for the local wired and wireless testing; encryption was left on during the wireless testing to more accurately simulate a real-world environment. The majority of network traffic was generated by the client to the server, with several other computers generating traffic through the router on the local network, but not to the client or server. This is an appropriate setup considering most institutions use a switched network, possibly a wireless network as well, with multiple users active at any given time. In our experiments there were two different scenarios:

1. Switched LAN – The data transfer occurs entirely over a 10/100 Ethernet network.
2. 802.11b WLAN – The data transfer occurs entirely over an 802.11b wireless network.

Several different configurations were used with multiple trials being run for each configuration.

Packets were queued using a Netfilter kernel module [26]. The packets were passed to user space if they were being sent out of a predetermined port, and placed into a separate queue with an appropriate release time set. If the packets were part of a watermark, the release time was set according to the watermark; if the packets were not part of a watermark, their release time was set to the current time. The head of the queue was continually checked to determine if the packet had reached its release time, and the packet was released if the current time was greater than the release time.

All network traffic was monitored using Tcpdump [27]. Traffic was generated by transferring a 2.97 MB text file using a simple sockets program written in C, resulting in the transfer of 2035 packets over the network for Figures 4 and 5. For Figures 6-9, the same sockets program was used to transfer 88,000 packets over the network. Each set of captured data was parsed to a flat file using a pcap parser. The files were then processed in Matlab to detect the watermark embedded within the data flow. Five hundred tests per network type were run without using the watermark to provide a baseline.[1] For each set of parameters used in Figures 6-9, ten trials were run per configuration, resulting in approximately 400 trials.

1 UDP traffic was generated as a CBR data flow (35Mbps)

5 Analysis

The balance of performance and security within large networks is crucial in maintaining the viability of that network. Corporate, university, government and military networks all require different levels of security and minimum tolerable levels of performance, leading to the need for scalable and adaptable solutions that do not prohibit performance. Using the *Covert Channel Model*, each network type receives a configurable and robust method to authenticate nodes for a minimal cost to the network. The cost of using these methods varies depending on the frequency of authentication, size of the watermark, and the amount of delay added. This cost is minimal over switched LAN and WLAN connections.

When adding a watermark, the majority of the data flow is uninterrupted (*Unwatermarked Delay*). Only the portion of the data flow containing the watermark adds delay (*Watermarked Delay*). Additionally, the number of times the watermark repeats increases the delay (*Watermarked Repetition Delay*), leading to the delay model (*Total Delay*).

unwatermarked_transmission_time = (packets_total – packets_watermarked) x (RTT/2)
high_delay = packets_watermarked @ λ x λ
low_delay = packets_watermarked @ ω x ω
watermarked_delay = high_delay + low_delay
watermarked_repititions_delay = total_repititions x watermarked_delay
transmission_time = unwatermarked_transmission_time + watermarked_repitions_delay

The delay added by the watermark depends on 4 variables: (1) The value of λ; (2) the value of ω; (3) the length of the watermark; and (4) the number of times the watermark is repeated. Both (1) and (2) are configurable and by minimizing these parameters the performance of the network is preserved. Both (3) and (4) are a function of the security required by the network.

The *Detector* attempts to minimize the values in (1) and (2). By using the values of first ten data packets sent, the *Detector* determines which type of network is being used and sets the rates accordingly. It determines the connection type based on the RTT time of the first ten packets. The distinction between a LAN and a WLAN is made based on the variance within the first ten packets. If a LAN is in use, the *Detector* sets $\lambda = (\varphi_{High} \times (RTT / 2))$ and sets $\omega = (\varphi_{Low} \times (RTT/2))$ with $\varphi_{High} = 46.5$ and $\varphi_{Low} = 15.5$. If a WLAN connection is in use the *Detector* sets the *Encoder* to "spike" the interface. When using a WLAN connection $\varphi_{High} = 1.84$ and $\varphi_{Low} = 1.0$; due to the comparatively high bandwidth of the LAN connection, larger normalizing parameters (φ_{High}, φ_{Low}) are required.

During testing, it was initially observed that large delays were required on WLAN networks to ensure that the entire watermark was received. When using limited bandwidth, such as that on a WLAN, packets may be queued within the interface before being transferred to the network. By adding a large delay ("a spike") prior to sending the watermark, the interface is allowed to clear its queue and thus transmit the watermark with greater accuracy and lower delays. Empirical testing shows that by "spiking" the interface, bandwidth over a WLAN connection is preserved far better than by simply increasing λ and ω. This will be discussed in more detail in Section VI.

While testing was done using both TCP and UDP traffic, we will focus our analysis on TCP. The primary reason for this is that UDP traffic is relatively well behaved (with no outside influences unlike TCP) and easily carries the watermark. This is shown in Fig. 3 where UDP traffic is sent over a WLAN connection (which is more volatile than a LAN connection) at a rate well below the rate determined to be optimal. During this trial, watermarks were detected at 100% over the entire duration. UDP traffic over a LAN connection showed similar results.

Ideally, after minimizing the delay, the watermark is transfered over the network and detected immediately. While this may be true with λ and ω set to a high value, it is not always the case. Network perturbations may cause alterations within the watermark leading to a false negative (no false positives were observed). In this case, multiple watermarks may be sent to authenticate the node. These incomplete watermarks are examined to determine their proximity to the correct watermark. Should they be within a tolerance level against false negatives (a predetermined value from the user) they will be considered the same as an ideal watermark. Additionally, care is taken such that delay values are not set too high (i.e., above the TCP timeout values), since packet retransmissions could also cause an increased number of false negatives.

Optimal values for encoding differ per network. To maximize throughput the desired value for ω is slightly above the normal sending rate. Desired values for λ are slightly but distinguishably above ω. While this varies depending on what network type is in use, testing within our testbed has shown that optimal values in our LAN are λ = 6ms and ω = 2ms and in our WLAN are λ = 22ms and ω = 16ms. This was determined empirically and is illustrated in Fig. 4 and Fig. 5. These values lead to a highly detectable solution that is still covert and minimizes the number of watermarks required for accurate detection.

Network performance is not greatly affected if λ and ω are set to relatively low values (but not the lowest possible values) and the number of watermarks contained within the authentication cycle is small. An authentication cycle represents the transmission of **one or more** watermarks. The transmission of several watermarks may be required to achieve an acceptable degree of detection accuracy. While it is certainly possible for a greater degree of performance to be achieved by setting λ and ω to lower values, doing so decreases the probability that the watermark will be accurately detected requiring a larger number of watermarks to be included within an authentication cycle.

The computational power required and the speeds at which the watermarks can be detected are also important factors. The Simple Threshold Method is a powerful and easily implementable method to detect the watermark. It only requires enough memory to buffer the current watermark, and minimal computations; as such, the Simple Threshold Method is faster than the Multi-Threshold Method. The drawback to using this method is that it has minimal intelligence, and can create a higher false positive rating if network conditions change. The Multi-Threshold Method provides a greater degree of robustness with lower false positive rates. However, it does require that the original time stamps be held in memory as well as the current watermark, and a slightly greater amount of computation to be done. Comparatively, the Multi-Threshold Method provides better protection against

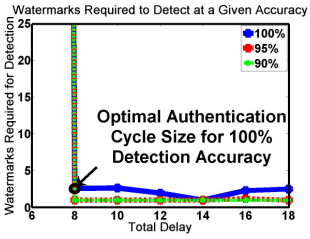


Fig. 4.

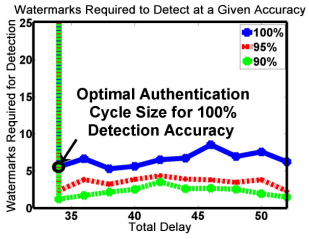


Fig. 5.

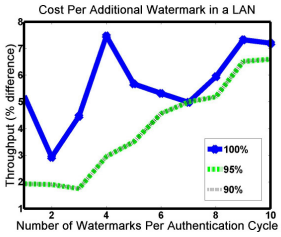


Fig. 6

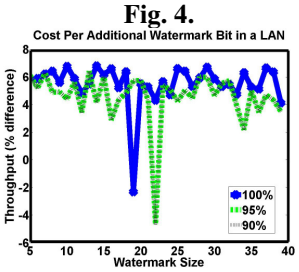


Fig. 7.

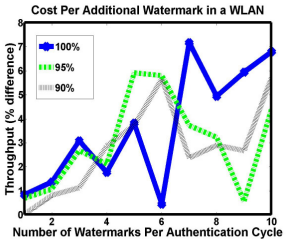


Fig. 8.

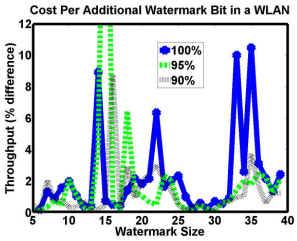


Fig. 9

Fig. 4 contains the results from trials used to empirically determine the optimal values for λ and ω, as well as the number of watermarks required per authentication cycle for different levels of detection accuracy in a LAN. Fig. 5 contains the results from trials used to empirically determine the optimal values for λ and ω, as well as the number of watermarks required per authentication cycle for different levels of detection accuracy in a WLAN. Figs. 6 and 8 show the percentage decrease in network throughput, compared to the baseline throughput, when the number of authentication cycles is increased in a switched LAN and WLAN respectively. Figs. 7 and 9 show the percentage decrease in network throughput, compared to the baseline throughput, when the length of a single watermark is increased in a switched LAN and WLAN respectively.

changes within network conditions. Due to space limitations our experimental data only provides analysis using the Simple Threshold method.

At the monitoring node each watermark is individually detected and compared to the true watermark to determine its accuracy. Each of these accuracy ratings are stored until enough watermarks have been received (or time has elapsed) to complete one authentication cycle. The maximum value of the stored accuracy ratings is selected and compared to the configured threshold. If this maximum is greater than the threshold the user is considered to be authenticated, otherwise the maximum value of the stored accuracy ratings is averaged with all previously detected maximum rates and this value is considered to be the user's current authentication value. The authentication value is compared to the configured threshold again to determine if the user is authenticated.

6 Performance Analysis

RTT values differ per network. During testing it was observed that RTT values for a switched LAN had a median value of 0.258ms and a mean value of 0.247ms. Within

a WLAN the RTT median value was 23.90ms and a mean value of 42.43ms. Using the *Detector*, the appropriate λ / ω delay times for a LAN and WLAN network are 6ms/2ms and 22ms/16ms respectively. As previously mentioned these values were determined empirically and are fixed in order to provide the performance analysis.

While certain institutions may require the highest level of security, the needs of other institutions may not be as stringent. The *Covert Timing Channel* method is designed to be a general method of authentication for institutions requiring various levels of security. As such, performance testing was done for several different levels of accuracy using different watermarks lengths and various numbers of watermarks included within a single authentication cycle. By using shorter watermark lengths and/or fewer watermarks per authentication cycle, institutions (such as educational facilities that do not require high levels of security) may authenticate nodes with a lower degree of accuracy without seriously impacting the performance of the network. Conversely, institutions requiring a greater degree of security may increase the length of the watermark and/or the number of watermarks per authentication cycle. Doing so allows nodes to be authenticated at a higher degree of accuracy while increasing the complexity required for watermarks to be detected at high levels of accuracy.

Testing on the switched LAN differed from that of the WLAN network in that the addition of a watermark was either fully detectable or not detectable at all up to 95% accuracy using an authentication cycle containing one watermark, after which an increase in the authentication cycle size (increased to three watermarks) was required to achieve 100% accuracy at lower rates. This is the result of extremely low latency between hosts. The "authentication watermarks" were fully detectable until 6ms/2ms, after which no watermarks were detected at all. As long as $\lambda > 5$ms and 1ms $< \omega \leq \lambda$-2ms, watermarks were generally 95% detectable with 0% error within the detection. The values of 6ms/2ms were chosen for the LAN because of their consistent ability to provide a great degree of detectability with minimal performance degradation (Fig. 4).

Our primary consideration during testing was the amount of overhead this model requires for accurate detection. Two configurable parameters were tested to provide a performance versus security comparison: (1) the cost to network throughput for different watermark lengths; and (2) the cost to network throughput for additional redundancy using authentication cycles. By increasing the length of the watermark and by adding additional authentication cycles, the security of the model is increased at a cost to the network throughput.

Fig. 6 represents the cost of increasing the number of watermarks contained within one authentication cycle. This figure provides results for 1 watermark, 20 bits (or packets) long, per authentication cycle, with one to ten authentication cycles being sent over the authentication period of 88,000 packets for 95% detection rate. To achieve 100% detection, 3 watermarks were used per authentication cycle using the same λ and ω values. The throughput of the watermarked flows are compared to the average of several hundred baseline transmissions in the same environment. Transmission of more authentication cycles caused a decrease in the throughput of the network. In our experiments it was noted that there was an average of 0.54% decrease

in throughput per additional authentication cycle transmitted, with a maximum performance penalty of 7.32% over our trials.

Fig. 7 represents the cost of increasing the watermark length, tested using 6 watermarks per authentication cycle for 100% detection, and 1 watermark per authentication cycle for 95% detection. The cost of increasing the watermark from a length of 5 to 40 is a total decrease in throughput of 7.5%, with each additional bit costing, on average, a 0.036% decrease in throughput.

Performance on a WLAN network had a greater degree of variance, primarily due to the nature of the network. Due to the bursty nature of a WLAN network, a higher proportional delay value was required for accurate detection. Testing was done at a detection accuracy of 90% (λ = 20ms), 95% (λ = 20ms), and 100% (λ = 22ms), with ω = 12ms (the optimal low value for our tested network) for each.

Fig. 8 represents the cost of increasing the amount of authentication cycles per authentication period on a WLAN network. Each authentication cycle contains 6/3/2 watermarks for 100%/95%/90% detection respectively, consisting of 20 bits (or packets). In our experiments it was noted that there was, on average, a 0.63% decrease in network throughput per additional authentication cycle added, with a maximum decrease of 7.17%.

Fig. 9 represents the cost of increasing the watermark length in a WLAN network. This figure contains a large degree of variance due to the nature of WLAN networks, but shows a general decrease in network throughput as the length of the watermarks are increased; with a larger amount of test data, we believe that this trend will smooth over time. On average adding an additional bit in a watermark on a WLAN network caused a 0.0435% decrease in throughput with, generally, an average maximum performance penalty of 8.54% when disregarding a single outlying value.

All figures, while showing the general trend expected, are comprised of a limited set of trials. It is our belief that the anomalies within each figure will smooth over a larger number of trials.

7 Conclusion and Future Work

While our specific application uses the *Covert Timing Channel* model to communicate authentication information, we investigate the use of the *Covert Timing Channel* model to transmit data in general. We show that the *Covert Timing Channel* method provides a secure, reliable, and cost effective process for which nodes may be authenticated within a network. Our model is highly scalable, easily adaptable and backwards compatible with existing technology, and provides excellent performance on switched LAN and WLAN networks. While the performance of this model is readily visible from the LAN and WLAN test results, we believe that this model could be extended for use over WAN networks as well; in our future work, we intend to show that this model will perform well over VPN connections. Additionally, we intend to investigate the effects of chaff packets on our method. Lastly, we shall apply our method to node authentication using real world traffic.

References

1. Cisco, NAC. http://www.cisco.com/en/US/netsol/ns466/networking_solutions_package.html
2. Thumann, M. Roecher, D. NAC@ACK: Hacking the Cisco Nac Framework. In the Proceedings of Black Hat Europe 2007.
3. Cisco Security Response: AAA Command Authorization By-Pass. http://www.cisco.com/warp/public/707/cisco-sr-20060125-aaatcl.pdf
4. Pyun, Y. J., Park, Y. H., Wang, X., Reeves, D. S., Ning, P. "Tracing Traffic through Intermediate Hosts that Repacketize Flows," in INFOCOM 2007. 26th IEEE International Conference on Computer Communications.
5. X. Wang and D. S. Reeves, "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Interpacket Delays," in Proc. of the 10th ACM conference on Computer and Communications Security (CCS), Oct. 2003, pp. 20–29.
6. X. Wang, S. Chen, and S. Jajodia, "Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet," in Proc. of the 12th ACM conference on Computer and Communications Security (CCS), Nov. 2005, pp. 81–91.
7. P. Peng, P. Ning, D. S. Reeve, and X. Wang, "Active Timing-Based Correlation of Perturbed Traffic Flows with Chaff Packets," in Proc. Of the 2nd International Workshop on Security in Distributed Computing Systems (SDCS), Jun. 2005, pp. 107–113.
8. X. Want, D.S. Reeves, P. Ning, and F. Feng. Robust Network-Based Attack Attribution through Probabilistic Watermarking of Packet Flows. Technical Report TR-2005-10, Department of Computer Science, NC State Univ., 2005.
9. P. Peng, P. Ning, and D. S. Reeves, "On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques," in Proc. of the 2006 IEEE Symposium on Security and Privacy (S&P), May 2006, pp. 334–349.
10. Y. Zhang and V. Paxson, "Detecting Stepping Stones," in Proc. of the 9th USENIX Security Symposium, Aug. 2000, pp. 171–184.
11. A. Blum, D. X. Song, and S. Venkataraman, "Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds," in Proc. of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID), Oct. 2004, pp. 258–277.
12. D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay," in Proc. of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID), Oct. 2002, pp. 17–35.
13. L. Zhang, A. Persaud, A. Johnson, and Y. Guan, "Stepping Stone Attack Attribution in Non-Cooperative IP Networks," Iowa State University, Tech. Rep. TR-2005-02-1, Feb. 2005.
14. Takahashi, T., Lee, W. "An Assessment of VoIP Covert Channel Threats," in Proc. Of SecureComm 2007, 3rd International Conference on Security and Privacy in Communication Networks.
15. X.Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao. On Flow Marking Attacks in Wireless Anonymous Communication Networks. In Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS), 2005.
16. S. Cabuk, C. Brodley, C. Shields, "IP Covert Timing Channels: Design and Detection," in the Proceedings of the 11th ACM conference on Computer and Communications Security, Oct. 2004.
17. Lampson, B. W. A Note on the Confinement Problem. Communications of the ACM 16, 10 (October 1973), pp. 613-615.
18. Rowland, C. H. Covert Channels in the TCP/IP Protocol Suite. First Monday 2, 5 (May 1997).
19. Fisk, G., Fisk, M., Papadopoulos, C., and Neil, J. Eliminating Stenagraphy in Internet Traffic with Active Wardens. In Information Hiding 2002 (2002), Springer, pp. 18-35.
20. Rutkowska, J.: The Implementation of Passive Covert Channels in the Linux Kernel. In: Chaos Communication Congress, Chaos Computer Club e.V. (2004).
21. route., alhambra. Project Loki. Phrack Volume 7, Issue 49, November 1996.
22. Moore, K. On the Use of HTTP as a Substrate. Tech. Rep., In Ternet Engineering Task Force, February 2002. RFC 3205.
23. Brinkhoff, L. GNU httptunnel. http://www.nocrew.org/software/httptunnel.html
24. Covert Channels Definition. http://en.wikipedia.org/wiki/Covert_channel
25. Stenography Definition. http://en.wikipedia.org/wiki/Stenography
26. Netfilter / IPTables. http://www.netfilter.org/
27. Tcpdump. http://www.tcpdump.org/