Fingerprint Embedding: A Proactive Strategy of Detecting Timing Channels

Jing Wang[1,2,⋆], Peng Liu[3], Limin Liu[1], Le Guan[1,2], and Jiwu Jing[1]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, CAS, Beijing, China
{jwang,lmliu,lguan,jing}@lois.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Pennsylvania State University, University Park, PA, USA
pliu@ist.psu.edu

Abstract. The detection of covert timing channels is notoriously a difficult work due to the high variation of network traffic. The existing detection methods, mainly based on statistical tests, cannot effectively detect a variety of covert timing channels. In this paper, we propose a proactive strategy of detecting covert timing channels. The basic idea is that a timing fingerprint is embedded into outgoing traffic of the to-be-protected host in advance. The presence of a covert timing channel is exposed, provided that the fingerprint is absent from the traffic during transmission. As a proof of concept, we design and implement a detection system, which consists of two modules for fingerprint embedding and checking, respectively. We also perform a series of experiments to validate if this system works effectively. The results show that it detects various timing channels accurately and quickly, while has less than 2.4‰ degradation on network performance.

Keywords: timing channel, covert channel, fingerprint embedding, intrusion detection system.

1 Introduction

Covert timing channel is a mechanism that exploits timing intervals of transmitted packets to convey sensitive information. Due to a large volume of data over the Internet, network traffic has become an ideal medium for stealthy communication. A cyber attacker can utilize this mechanism for various purposes, e.g., exfiltrating secrets [25], launching DDoS attacks [13], and tracking network flows [30]. Under the cover of massive overt traffic, it is generally hard to reveal

⋆ This work was supported by National Natural Science Foundation of China Grant 70890084/G021102 and 61003274, Strategy Pilot Project of Chinese Academy of Sciences Sub-Project XDA06010702, National High Technology Research and Development Program of China (863 Program, No.2013AA01A214), ARO W911NF-09-1-0525 (MURI), NSF CNS-0905131, NSF CNS-0916469, and AFOSR W911NF1210055.

S. Qing et al. (Eds.): ICICS 2013, LNCS 8233, pp. 229–244, 2013.

the presence of a covert timing channel. Sometimes, an attacker creates one in a victim machine even with no additional traffic, only by manipulating the timings of the existing traffic, under which circumstance the system administrator is practically unconscious of it. This kind of channels, referred to as passive timing channels, are more difficult to be detected than active ones.

The detection of covert timing channels is acknowledged as a difficult work all along. In the current literature, detection methods are mainly based on statistical tests, which differentiate covert traffic with irregular statistical properties from legitimate traffic. Empirically, these detection methods are over-sensitive to the high variation of network traffic, and thereby causes a unacceptable false alarm rate. Due to the inherent limitation, statistical-test-based detection methods are strongly dependent of encoding techniques applied in timing channels, and only accurate in detecting a specific timing channel. Furthermore, if covert traffic is designed to be statistically approximate to legitimate traffic, these methods are unable to detect it. In addition, the extraction of statistical properties requires large test samples. If a covert channel exists, a large amount of sensitive information (e.g., possibly, encryption keys or passwords) has already been leaked before it is detected. In brief, previous detection methods have limitations in detection effectiveness.

In this paper, we propose a proactive strategy of detecting covert timing channels. The basic idea is that a secret fingerprint is embedded into outgoing traffic of the to-be-protected host in advance. Meanwhile, a detector, which is located in the transmission path of the protected traffic, checks if the traffic still retains the fingerprint. The presence of a covert timing channel is exposed, provided that the fingerprint is absent.

To demonstrate the feasibility of the proactive strategy, we design and implement a detection system targeting passive timing channels, whose covert encoder resides on the intermediate routers/gateways or low-layer network protocol stack of the victim machine. The detection system consists of two modules: `ADAPTOR` and `CHECKER`. `ADAPTOR` is installed on the machine in protection, and the purpose is to embed a specific fingerprint shared with `CHECKER` into each flow of network traffic; `CHECKER` is located on the intrusion detection system (IDS) to examine the presence of the fingerprint in passing traffic. To evaluate the performance of our detection system, we also conduct a series of experiments to validate if the detection system works effectively. In the experiments, it can successfully detect existing timing channels and raw legitimate traffic with no fingerprint, which indicates that the performance of our detection method is independent of channel encoding methods. Moreover, the results show that our system can detect covert traffic not only accurately but also quickly, while has little influence on network performance.

The rest of the paper is organized as follows. In Section 2, we discuss the related work in timing channels. After that, we introduce the communication scenario, the system model and the adversary model. In Section 4, we describe the structure of our detection system. Then we validate the effectiveness of this system. Finally, we conclude the paper.

2 Related Work

The timing of some events can leak secret information in a manner that compromises security properties. In side timing channels, attackers exploit the fact that the time for executing a cryptographic algorithm is data dependent. Specifically, key information can be extracted through precise measurements of the time [17]. Side-channel timing attacks have also been proved to be practical over a network [4,3,7]. In addition, it has been shown that such attacks are effective not only on cryptosystems but also in other scenarios [8,27]. However, the focus of this paper is on deliberate information leakage from timing channels established by a premeditated adversary rather than unintentional leakage from side channels.

Inter-packet delays in network traffic can be used as a medium for data stealth transfer. This kind of communication channel, which is not intended for data transfer, is usually called "covert channel" [19]. Compared with network storage channel, which exploits the redundancy of network protocols, timing channel is more stealthy due to the cover of varying overt traffic.

The recent literature is not lack of practical exploitation of covert timing channels. Since the first IP covert timing channel came into being [6], channel exploitation has made great progress over the years. In [25], Shah et al. designed a device called Jitterbug, which can extract typed information and leak it through a timing channel. This is a typical example of real-world network timing channels. Gianvecchio et al. [10] proposed a model-based covert timing channel, which mimics the distribution of legitimate traffic. Liu et al. [20] introduced spreading codes to encode messages for the purpose of increasing the robustness of timing channel. Sellke et al. [24] proposed the "L-bits to N-packets" encoding scheme in order to build a high-capacity covert timing channel. In general, the research on network covert timing channel exploitation has been done from three aspects: undetectability [5,10,28,18], robustness [14,20], capacity [2,34,32].

Covert timing channel can be categorized into two types: active and passive. In an active channel, the sender generates additional network packets to embed covert information and needs to compromise the host for the total control over it. While in a passive channel, the sender just compromises the I/O device [25] or low-level network protocol stack [33], or resides in the middle nodes (e.g., routers or gateways) [21,34], manipulating the existing network traffic. In general, it is harder to discover passive channels due to their parasitism on overt ones. Additionally, in terms of encoding techniques, passive channels usually maintain the inter-packet dependencies of legitimate traffic, and thus, are hardly detectable. The detection of them has become a challenging work.

There is another form of passive timing channel, i.e., network flow watermark, intended for packet traceback. Wang et al. [31] develop a watermark, aiming to correlate "stepping stones", by means of sightly changing the timing of flow-based packets. This technique can also be used to correlate encrypted VoIP calls even in anonymous networks [30].

To eliminate the threat from covert timing channels, researchers have proposed a series of solutions: fuzzy time [15], jammers [11], pump [16], etc. The fundamental idea among them is to insert random interrupts in order to lower

the channel capacity boundary. However, network suffers performance degradation therefrom. In addition, covert timing channel is still able to exist but with a poor bandwidth. By comparison, it might be preferable to adopt passive traffic analysis, namely channel detection, which aims to differentiate covert traffic from normal traffic. The research on this field has attracted a lot of attention in recent years, and meanwhile, a series of detection techniques have been developed [6,2,22,9]. The most noteworthy among these, supposedly, is the entropy-based method [9]. This method utilizes entropy and conditional entropy to detect the anomaly in first-order and high-order statistics, respectively. Compared with the others, empirically, it has a better performance in detecting various covert timing channels. Even so, it still can be defeated by some mimicry-based encoding techniques [28,18].

3 Preliminaries

3.1 Communication Scenario

A covert timing channel consists of a covert encoder sending secrets by modulating network event timings, and a decoder extracting secrets by observing the timings. In reality, the covert encoder is not necessarily the sender of network traffic. Specifically, it can be located in the transmission path, and manipulates the delays of passing-by packets. Sometimes, an attacker may choose to create such a passive channel instead of an active one, the reasons of which include: 1) sometimes the creation of a passive channel does not require a compromised host. 2) the generation of additional traffic is prone to exposing the presence of a covert channel, whereas a passive channel can avoid this risk. There has been recognized difficulty in mitigating the security threat from such channels. In this paper, we target passive channels to demonstrate the feasibility of our proactive strategy. For the sake of clarity, we elaborate that, in the scenario of covert communication we are concerned with, as shown in Figure 1, the covert encoder can reside on the intermediate routers/gateways near the overt sender, or even on the compromised output device or low-layer network protocol stack of the same machine as the overt sender. We note that the scenario of Jitterbug [25], which compromises an input channel, is not in the application scope of our proof-of-concept implementation.

3.2 System Model

In a general system model of network covert communication, there is another role besides a covert encoder and decoder, that is, a network warden, residing between the covert encoder and decoder and monitoring the passing messages [12]. There are mainly two types of wardens in the literature of covert timing channel:

– A *passive* warden can detect the presence of covert timing channels, but cannot alter the existing traffic.

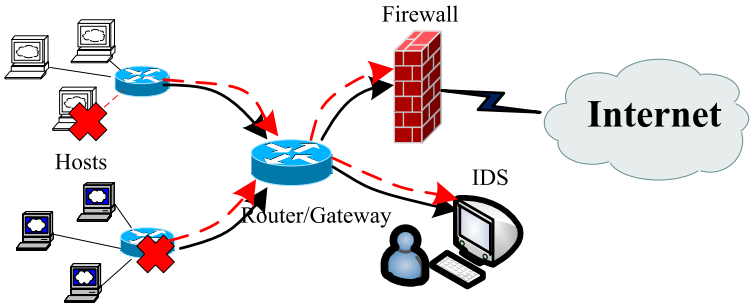

Fig. 1. Communication Scenario

– An *active* warden can alter the timings of existing traffic to eliminate or disrupt covert timing channels.

In this paper, we introduce a new type of warden, which is an *active* one but intended for detection. Specifically, the warden embeds a specific timing fingerprint into the newly generated traffic by slightly altering inter-packet delays, and meanwhile, he is located in the transmission path and detects if the transmitted packets still retain the specific fingerprint.

3.3 Adversary Model

In some cases of passive channels, the location of an adversary is physically separated from the overt sender, e.g., intermediate routers/gateways, so the adversary is unable to directly access secrets in the victim machine. On the other hand, an adversary might reside in some important components of the victim machine, e.g., the output device or low-layer network protocol stack, and thus, have an opportunity to steal secrets about fingerprint information. To consider the worst situation, we assume the adversary has the ability to compromise the host; then he can grab desirable information.

We also assume that virtualization technology is implemented in the to-be-protected machine. Such assumption is realistic because this technology has been widely applied in personal computers, servers, and workstations. The self-protection of secrets about specific fingerprints and fingerprint embedding code can be addressed by applying the SIM [26] framework. Even if an adversary compromises the OS kernel, he is unable to know about the fingerprints or disable the code.

4 Detection System

The detection system consists of two modules: `ADAPTOR` and `CHECKER`. `ADAPTOR`, which is installed on the to-be-protected machine, embeds a specific fingerprint

into each flow of network traffic. CHECKER, installed on the IDS, which is located on the edge of the LAN, checks if the passing traffic still retains the fingerprint. Figure 2 shows the overview of our detection system.

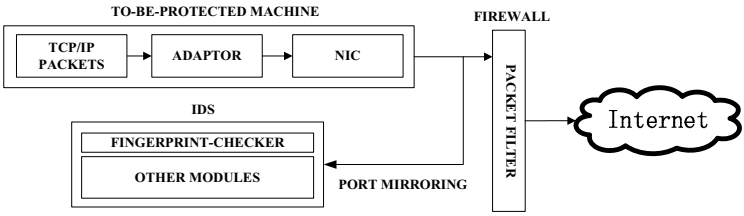

Fig. 2. System Overview

In the practical implementation, ADAPTOR and CHECKER agree a priori on the knowledge about fingerprint patterns, including the specific encoding/decoding method and the involved secrets (e.g., sensitive parameters or random number seeds). ADAPTOR can be installed on multiple to-be-protected machines in a LAN, but the fingerprint patterns may be distinct from each other. CHECKER is only required to be installed on the boundary between the LAN and the outside world, and thus, is responsible to check all the traffic generated from protected machines in the LAN. CHECKER should maintain a list of services in protection and corresponding fingerprint patterns in use. The pair of each ADAPTOR and CHECKER constitutes a trusted transmission path. If the timing interval pattern of a flow deviates from the expected one, a covert timing channel probably exits in the corresponding transmission path. A warning will be given to the security administrator once CHECKER detects such traffic anomaly. The ADAPTOR and CHECKER modules are detailed in the following subsections.

4.1 ADAPTOR

The main task of the ADAPTOR module is to change the timing characteristic of the original traffic, which is intended to leave its fingerprint on the transmitted traffic. As an exemplary implementation, we place ADAPTOR between the OS kernel and network interface card. ADAPTOR intercepts TCP/IP packets just generated from the OS. The original traffic is stored temporarily in a buffer, waiting to be forwarded at an elaborately planned time. ADAPTOR classifies the traffic into individual flows based on protocol, source and destination port and IP address, and then calculates intervals between adjacent packets of each flow. The transmission time of each packet is determined according to the original intervals and the fingerprint pattern. ADAPTOR finally forwards the manipulated packets to network interface card. Figure 3 shows the workflow of ADAPTOR.

Fingerprint Encoding. There is a basic principle regarding fingerprint encoding, that is, the timing of the original traffic cannot be altered a lot. If the

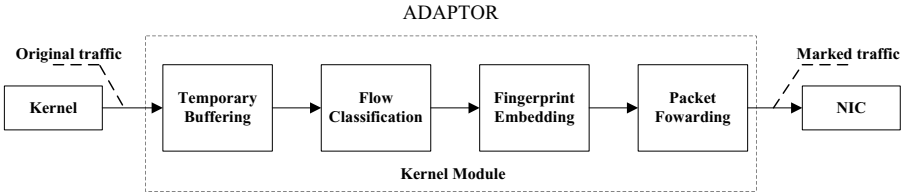

Fig. 3. The workflow of ADAPTOR

packets are delayed for a long time, the network performance will suffer penalty seriously. We note that, this is similar to the negative effect of current disruption techniques, which reduces covert channel capacity by adding random delays to traffic. For this reason, a slight change in packet intervals is only allowed when fingerprint information is encoded into the original traffic.

We develop a simple fingerprint encoding scheme for the detection system. The sequence of original times when packets are delivered from the kernel is denoted as $\{t_1, ..., t_n\}$ here. To mark the traffic generated from the to-be-protected machine, we add a slight delay noted by τ_i to each element of the sequence. So the new sequence of altered times when packets are forwarded by ADAPTOR turns out to be $\{t'_1, ..., t'_n\}$, where $t'_i = t_i + \tau_i$. The fingerprint of ADAPTOR is marked on the resulting inter-packet delays $IPD_i = t'_{i+1} - t'_i$, which satisfies the following equation:

$$IPD_i \ mod \ \omega \ = \ 0$$

where ω is a time parameter, referred to as the fingerprint pattern.

We can infer that the delay τ_i is less than ω all the time. Therefore, the parameter ω determines the maximum delay to packet transmission time, and hence, directly affects the network performance.

An example is given below to help the understanding of the fingerprint encoding scheme. We assume $\omega = 1000us$, and the sequence of original intervals is $\{375, 17790, 3889, 8456, 55322\}us$. To make each interval be an integral multiple of ω, ADAPTOR adds a delay to each packet, and consequently, the sequence of altered intervals turns out to be $\{1000, 18000, 4000, 9000, 56000\}us$.

Influencing Factors. As described above, the delays manipulated by ADAPTOR, which are bounded from above by the parameter ω, cannot be too long in consideration of user experience. On the other hand, if ω is small, the fingerprint will be wrongly decoded on the module CHECKER side although no intentional change occurs during transmission. There are several factors influencing the accuracy rate of fingerprint decoding.

The major one is jitter. In the context of computer networks, jitter, also referred to as packet delay variation, represents an average of the deviation from the packet mean latency over a network. From another perspective, it indicates that a network has inconstant latency, and there always exists a difference between an inter-packet delay on the sender side and that on the receiver side.

A packet forwarded later may even reach at an earlier time due to the large network jitter. The timing fingerprint cannot avoid the disturbance from network jitter during packet transmission.

To ensure CHECKER decodes the fingerprint correctly under the existence of jitter, ω must be at least two times larger than the largest jitter in a LAN. To demonstrate this, we assume that ADAPTOR sends the i-th packet at t'_i time, and the packet is transmitted with an average delay of θ and a random jitter of γ which is bounded by the inequality $-\gamma_{max} \leq \gamma \leq \gamma_{max}$, where γ_{max} denotes the longest jitter. Then the CHECKER observes the very packet at $t + \theta + \gamma$. Figure 4 shows two extreme cases in which the observed intervals are changed by $2\gamma_{max}$. Therefore, to counter network jitter, the parameter ω should be much larger than $2\gamma_{max}$.

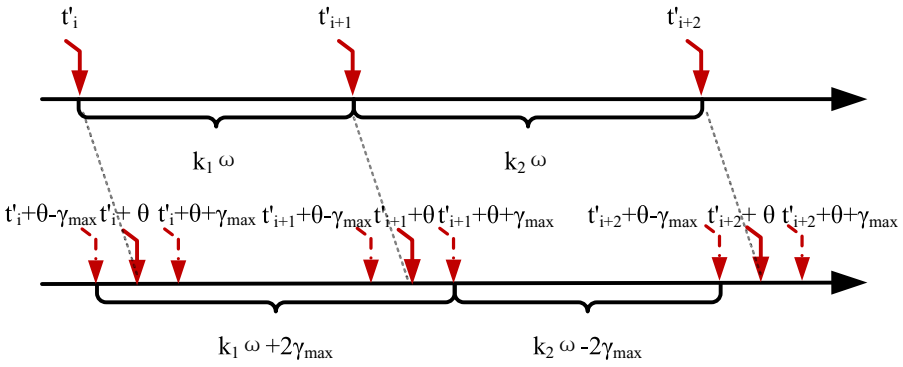

Fig. 4. Jitters involved in transmission

In addition, clock skew is another influencing factor. As a matter of fact, the clock frequencies of the protected machine and IDS where two modules are installed respectively may not be completely identical. Although the difference is rather small, it poses a non-ignorable obstacle to the precise fingerprint embedding. Empirically, there is indeed a *linear* relationship between the timing shifts and intervals. The asynchrony can be addressed by estimating the timing ratio between the clock of the protected machine and that of the IDS as follows. The protected machine transmits two packets with an interval of θ, and then the corresponding interval θ' in us is obtained on the IDS. The ratio is simply θ/θ'. The timing intervals observed in the IDS will be finally multiplied by this ratio so that the clocks are adjusted to be synchronous.

4.2 CHECKER

CHECKER is used in conjunction with ADAPTOR. It sits on the IDS of a LAN, and takes charge of all outbound traffic from protected hosts in the LAN. Firstly, the IDS monitors traffic of inside network via a mirroring port which carries a copy

of outbound traffic. CHECKER then singles out the traffic in protection according to the maintained list. Sequently, since fingerprint embedding is flow based, CHECKER should classify the picked traffic into individual flows based on the traffic 5-tuple. Each flow will be examined for the presence of the expected fingerprint. If the fingerprint is absent, it implies that there is probably a covert timing channel in the transmission path. Figure 5 shows the workflow of CHECKER.

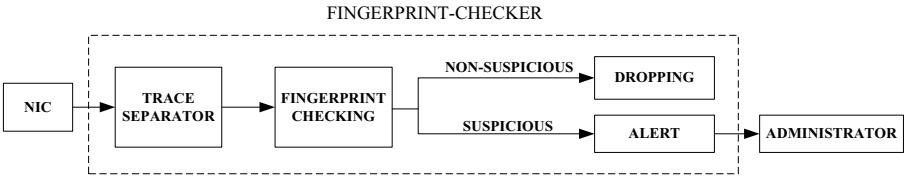

Fig. 5. The workflow of CHECKER

Fingerprint Decoding. CHECKER takes the reverse operation of fingerprint embedding to determine if packet intervals retain the expected fingerprint pattern. For decoding, we assume the sequence of packets reaches the IDS at the times of $\{T_1, ..., T_n\}$. Each inter-packet delay is then denoted as $\widehat{IPD}_i = T_{i+1} - T_i$. To remove the effect of network jitter detailed in Section 4.1, the decoding of the fingerprint allows each interval has small fluctuation. The decoding algorithm is given as follows: if $-\beta \leq \widehat{IPD}_i \leq \beta \ (mod\ \omega)$, where β is a tolerance parameter, then the fingerprint is identified as present; otherwise, the fingerprint is identified as absent. Obviously, the selection of β depends on the amount of network jitter in the LAN. To tolerate the worst cases, β should be no less than the double of the largest jitter γ_{max}.

In addition, to counter bad network conditions, we utilize a tolerant mechanism for determining whether a covert timing channel exits. If the fingerprint pattern is absent in an interval, the corresponding packet is flagged as illegitimate. When an illegitimate packet comes along, a burglar alert will not be immediately sent out. Actually, only a lot of illegitimate packets during a period will trigger the IDS to send an alert to system administrator. We use the density of illegitimate packets to indicate the health condition of the host in protection. More specifically, we record the number of illegitimate packets among the previous 10 ones. If this number exceeds a threshold, the trusted transmission path is identified to contain a covert timing channel.

4.3 Security Enhancement

There are some potential attacks against the naive fingerprint encoding method. To mitigate these attacks, two security-enhanced schemes based upon the naive one are presented hereinafter.

Potential Attacks. The fingerprint encoding method has some drawbacks that can be exploited by an attacker to forge the fingerprint pattern in use or evade fingerprint checking. We discuss potential attacks in the following paragraphs.

1. *Forging fingerprint* After the fingerprint is embedded, packet intervals turn into be multiples of the parameter ω. An attacker located on a router inside the LAN can eavesdrop the traffic delivered from the host in protection and extract the value of ω. If ω is not changed constantly, the attacker is able to forge `ADAPTOR`'s fingerprint and embed it into covert traffic. `CHECKER` will identify covert traffic with forged fingerprint as legitimate, and thereby, covert traffic can pass through `CHECKER` undetected.
2. *Invalidating censorship* The parameter ω is generally much smaller than the mean intervals of legitimate traffic. Some simple encoding schemes of timing channels encode bit information in fixed and relatively long intervals, e.g., IP-PCTC [34] transmits 0-bit by a 5ms interval and 1-bit by a 12ms interval. Thus, the intervals of covert traffic happen to be multiples of ω. If those channels can manipulate packet delays exactly on the millisecond, `CHECKER` also fails to detect them.

Enhanced Schemes. One solution is to constantly change the parameter ω. `ADAPTOR` and `CHECKER` are assumed to negotiate a set of ω values in advance. For instance, there are 10 elements in the set, ranging from 1000us to 1900us with the gradient of 100us. The parameter values are indexed from 0 through 9, respectively. We also assume `ADAPTOR` and `CHECKER` share a pseudo-random sequence of integers that range from 0 to 9. The random seed and the set of values used by `ADAPTOR` can be protected by the hypervisor in the host from being stolen, as described in Section 3.3. For each inter-packet delay, ω value to be used in the encoding algorithm is determined according to the newly generated random number. Meanwhile, the same process is applied to decode the fingerprint.

Another solution is to introduce another more secure parameter in the encoding algorithm. We assume `ADAPTOR` and `CHECKER` share a pseudo-random bit stream $\{r_1, ..., r_n\}$. The random seed used by `ADAPTOR` can protected in the same way as hereinbefore. The encoding algorithm turns into be as follows:

$$IPD_i \bmod \omega = \begin{cases} 0, & \text{if } r_i = 0; \\ \frac{\omega}{2}, & \text{if } r_i = 1. \end{cases}$$

where IPD_i is the resulting inter-packet delay as before, and ω is also the same as before. To decode the fingerprint, the following algorithm is used:

$$r_i = \begin{cases} 0, & \text{if } -\beta \leq \widehat{IPD}_i \leq \beta \ (mod\ \omega); \\ 1, & \text{if } \omega/2 - \beta \leq \widehat{IPD}_i \leq \omega/2 + \beta \ (mod\ \omega); \\ invalid, & \text{otherwise.} \end{cases}$$

We note that this enhanced scheme is inspired by Jitterbug [25]. This solution can be combined with the former to achieve more secure properties.

5 Implementation and Evaluation

We implemented a prototype in a real network environment to validate the effectiveness of our detection strategy. In this section, we first describe the implementation details at a high level. Then, we measure the maximum network jitter, based on which we determine the parameter ω and β. Finally, our detection test is performed against two existing timing channels: MBCTC [10] and Jitterbug [25]. Furthermore, we test if the detection method works against legitimate traffic that has no fingerprint embedded.

The evaluation criterions include false positive rate, false negative rate, detection latency, and performance penalty. An ideal detection method achieves both low false positive and false negative rate, while has small detection latency and little influence on system performance. Generally, conventional detection tests are unable to satisfy the first three due to their inherent limitation that comes with statistical tests. The purpose of our proposed idea is to conquer this limitation, thereby achieving a high performance on detection regardless of encoding techniques used in covert timing channels.

5.1 Implementation Details

Our design goal is to be effective in detecting passive timing channels as much as possible. We implemented `ADAPTOR` in Linux environment as a kernel module, which can embed its fingerprint into a newly-generated packet. Netfilter [23] hooks were utilized to intercept packets. For fingerprint encoding, we used udelay() function to add delays to the original packets. This is because the manipulation of intervals when embedding a fingerprint needs to be very precise. Additionally, we disabled Large Segment Offload feature, which alleviates the burden of operating system by allowing it to assemble large packets and by transferring the task of disassembling large packets into smaller segments to NIC. This technology has been widely deployed in Linux since kernel 2.6.18. The host in our prototype runs a Linux OS with kernel version 2.6.35.

We implemented `CHECKER` as a module of an IDS in Windows environment. The IDS and the hosts in protection are located within the same LAN. For this reason, the switch of the IDS is configured to mirror all the outbound traffic to the IDS port. CHECKER only need to monitor the outbound traffic and record packet arriving times. This module was implemented using WinPcap [1].

5.2 Parameter Determination

In order to counter the ubiquitous existence of network jitter, the parameters ω and β are introduced in fingerprint encoding and decoding algorithms (detailed in Section 4.1 and 4.2). More specifically, β is set to tolerate bad network conditions and hence no less than jitter, while ω should be much larger than β so as to have a high detection effectiveness. From a statistical perspective, packet intervals with no fingerprint modulo ω uniformly fall in the range $[-\frac{\omega}{2}, \frac{\omega}{2}]$, so the probability that they fall in the tolerant range $[-\beta, \beta]$ is $\frac{2\beta}{\omega}$. When this

value is small, the situation that an illegitimate packet is wrongly identified as legitimate occurs with a low probability. We set the ratio of $\frac{2\beta}{\omega}$ to be $\frac{1}{20}$. For example, when ω is $400us$, the fingerprint pattern is valid if an observed interval is in $[k * \omega - 10, k * \omega + 10]us$ $(k \in N_+)$.

Since ADAPTOR and CHECKER both reside in the LAN, the variation of network latency is intuitively rather small. To investigate the real jitter in a LAN, we conducted a series of experiments as follows. We sent from ADAPTOR 4 test sets of packets, whose intervals are multiples of $50us, 200us, 800us$, and $1000us$, respectively. Each test set has 1027 packets (1026 intervals). On the CHECKER side, we calculated the mean and standard deviation of observed intervals modulo the respective parameter value, as shown in Table 1. Note that the count represents the number of intervals beyond the tolerant range. We can observe that when $\omega = 800us$, the false alarm rate is very low, only 0.19%, which is acceptable. In fact, approximately 85 percent of intervals modulo $800us$ gather within the range $[-10, 10]us$. We choose $\omega = 800us$ and $\beta = 20us$ in the following experiments.

Table 1. Timing Jitters

Parameter/us	Mean/us	StdDev/us^2	Count	Ratio
50	-0.3655	8.09576	908	88.50%
200	0.04191	7.9225	140	13.65%
800	0.14352	8.1617	2	0.19%
1000	0.16179	10.19637	2	0.19%

5.3 Detection

Real Timing Channels. To investigate the effectiveness of our detection system, we performed the detection test against existing timing channels: MBCTC [10] and Jitterbug [25]. We only concern about their encoding methods. We implement their encoding methods with ADAPTOR disabled, which means no fingerprint is embedded in their traffic. For MBCTC, we utilize HTTP traffic, which is extracted from NZIX-II data sets [29], as the modeling objective. We fit a set of 100 packets to a model and use the model to generate covert traffic. For Jitterbug, we use SSH traffic, also from NZIX-II data sets, to transfer covert information. The timing window is set at 20 milliseconds, and a pseudo-random sequence of integers is used to smooth out interval patterns, as suggested by Shah et al. [25].

As illustrated in Figure 6, the number of illegitimate packets among the previous 10 ones rises rapidly when the transmission of covert packets starts. We set the threshold to be 5. This indicates if the number exceeds 5, the traffic will be identified as illegitimate, and then an alert will be sent to the security administrator. Based on this given threshold, the detection latency and effect is given in Table 2. We can see that no more than 7 packets (6 intervals) have been sent out before the presence of a timing channel is detected. In addition, the test maintains both 0% false positive rate and false negative rate. Our detection method

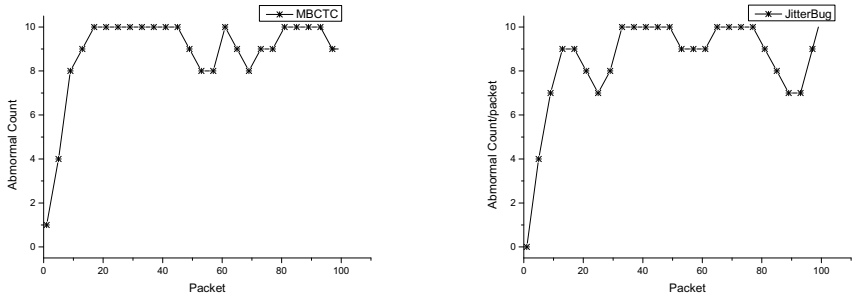

Fig. 6. Detection against real timing channels

achieves much higher detection performance compared to statistical-test-based detection methods, which not only require thousands of packets to analyze traffic behavior, but also have over sensitivity to the high variation of traffic [9].

Table 2. Detection latency and effect

	MBCTC	JitterBug
Latency/interval	6	6
Information Disclosure/bit	6	6
False Positive Rate	0%	0%
False Negative Rate	0%	0%

Raw Legitimate Traffic. To validate the effectiveness of our detection system further, we conducted the detection test against raw legitimate traffic which is legitimate but with no fingerprint. We replayed HTTP traces from NZIX-II data sets. During the experiment, we enabled and disabled `ADAPTOR` alternately, as shown in Figure 7. After `ADAPTOR` is disabled at the 16th packet, the abnormal count reaches up to 10 immediately and then stays around there. When `ADAPTOR` is enabled again at the 56th packet, the abnormal count decreases gradually to 0. This experimental result shows that even legitimate traffic can be detected by this test as long as it has no fingerprint. From another aspect, it indicates the effectiveness of our detection method is independent of encoding methods. That is to say, no matter how approximate covert traffic can be to legitimate overt traffic, this method is still successful in detecting it.

5.4 Network Performance

Since packets are delayed when `ADAPTOR`'s fingerprint is embedded, network performance degrades more or less. To estimate the practical performance penalty,

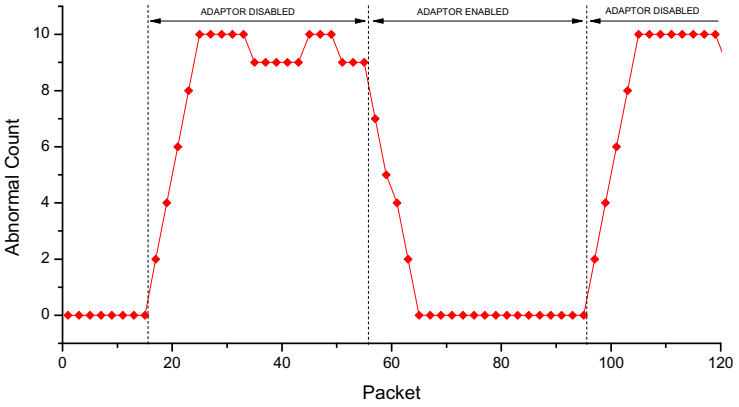

Fig. 7. Detection against raw legitimate traffic

the test machine sent 2 sets of HTTP packets, which are selected from the previous traffic. The first set consists of low-density traffic, while the second one is highly dense. We calculated the difference between the total transmission time with `ADAPTOR` and that without `ADAPTOR`. The penalty is set as the ratio of the difference and the transmission time without `ADAPTOR`, as given in Table 3. We consider the performance penalty of no more than 2.4‰ to be acceptable.

Table 3. Performance Penalty

Test set	Number of intervals	Mean	Sum(without adaptor)	Sum(with adaptor)	Penalty
1	10000	0.91s	9070.45s	9074.59s	0.46‰
2	10000	0.17s	1681.52s	1685.56s	2.40‰

6 Conclusion and Future Work

We introduced a proactive strategy of detecting covert timing channels. The basic idea is that a timing fingerprint is embedded into outgoing traffic of the to-be-protected host in advance. If a covert timing channel exits in the transmission path, the fingerprint will probably be disrupted, and thereby the presence of the timing channel can be detected. We described our detection system, a proof-of-concept implementation aiming at passive timing channels. This system consists of the `ADAPTOR` module and `CHECKER` module: the former is located on the machine in protection, and the purpose is to embed a specific fingerprint into each flow of network traffic; the latter resides on the IDS of the same LAN to examine if the passing traffic retains the fingerprint.

We then applied our detection system to detect covert timing channels. The experimental results show that it can detect various existing timing channels

accurately and quickly. Even raw legitimate traffic with no fingerprint is also detectable by this system. This indicates that the performance of our detection techniques is independent of channel encoding methods. Finally, we showed that the penalty on network performance is acceptably small.

The arms race between covert timing channel design and detection techniques has been ongoing. In this paper, we suggested some simple ideas to the design of detection systems. We hope our proactive techniques can open a new perspective for researchers. To further improve the proactive detection scheme, we plan to investigate this direction in the future.

References

1. Winpcap: The windows packet capture library, `http://www.winpcap.org`
2. Berk, V., Giani, A., Cybenko, G., Hanover, N.: Detection of covert channel encoding in network packet delays. Tech. Rep. TR2005-536, Dartmouth College, Computer Science, Hanover (2005)
3. Brumley, B.B., Tuveri, N.: Remote timing attacks are still practical. In: Proceedings of the 16th European Symposium on Research in Computer Security, pp. 355–371 (2011)
4. Brumley, D., Boneh, D.: Remote timing attacks are practical. In: Proceedings of the 12th Conference on USENIX Security Symposium (2003)
5. Cabuk, S.: Network covert channels: design, analysis, detection, and elimination. PhD thesis (2006)
6. Cabuk, S., Brodley, C., Shields, C.: Ip covert timing channels: design and detection. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 178–187 (2004)
7. Crosby, S.A., Wallach, D.S., Riedi, R.H.: Opportunities and limits of remote timing attacks. ACM Transactions on Information and System Security 12(3), 17 (2009)
8. Felten, E.W., Schneider, M.A.: Timing attacks on web privacy. In: Proceedings of the 7th ACM Conference on Computer and Communications Security, pp. 25–32 (2000)
9. Gianvecchio, S., Wang, H.: Detecting covert timing channels: an entropy-based approach. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 307–316 (2007)
10. Gianvecchio, S., Wang, H., Wijesekera, D., Jajodia, S.: Model-based covert timing channels: Automated modeling and evasion. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 211–230. Springer, Heidelberg (2008)
11. Giles, J., Hajek, B.: An information-theoretic and game-theoretic study of timing channels. IEEE Transactions on Information Theory 48(9), 2455–2477 (2002)
12. Handel, T., Sandford, M.: Hiding data in the OSI network model. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 23–38. Springer, Heidelberg (1996)
13. Henry, P.: Covert channels provided hackers the opportunity and the means for the current distributed denial of service attacks. CyberGuard Corporation (2000)
14. Houmansadr, A., Borisov, N.: CoCo: Coding-based covert timing channels for network flows. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 314–328. Springer, Heidelberg (2011)
15. Hu, W.M.: Reducing timing channels with fuzzy time. In: IEEE Symposium on Security and Privacy, pp. 8–20 (1991)

16. Kang, M., Moskowitz, I.: A pump for rapid, reliable, secure communication. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 119–129 (1993)
17. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
18. Kothari, K., Wright, M.: Mimic: An active covert channel that evades regularity-based detection. Computer Networks (2012)
19. Lampson, B.: A note on the confinement problem. Communications of the ACM 16(10), 613–615 (1973)
20. Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A.-R., Schulz, S., Katzenbeisser, S.: Hide and seek in time — robust covert timing channels. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 120–135. Springer, Heidelberg (2009)
21. Lucena, N.B., Pease, J., Yadollahpour, P., Chapin, S.J.: Syntax and semantics-preserving application-layer protocol steganography. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 164–179. Springer, Heidelberg (2004)
22. Peng, P., Ning, P., Reeves, D.: On the secrecy of timing-based active watermarking trace-back techniques. In: IEEE Symposium on Security and Privacy (2006)
23. Russell, R., Welte, H.: Linux netfilter hacking HOWTO (2002), `www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html`
24. Sellke, S., Wang, C., Bagchi, S., Shroff, N.: TCP/IP timing channels: Theory to implementation. In: INFOCOM 2009, pp. 2204–2212 (2009)
25. Shah, G., Molina, A., Blaze, M.: Keyboards and covert channels. In: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15 (2006)
26. Sharif, M.I., Lee, W., Cui, W., Lanzi, A.: Secure in-VM monitoring using hardware virtualization. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 477–487 (2009)
27. Song, D.X., Wagner, D., Tian, X.: Timing analysis of keystrokes and timing attacks on SSH. In: Proceedings of the 10th USENIX Security Symposium, vol. 2, p. 3 (2001)
28. Walls, R., Kothari, K., Wright, M.: Liquid: A detection-resistant covert timing channel based on IPD shaping. Computer Networks 55(6), 1217–1228 (2011)
29. WAND Research Group: Waikato internet traffic storage, `http://wand.net.nz/wits/nzix/2/`
30. Wang, X., Chen, S., Jajodia, S.: Tracking anonymous peer-to-peer voip calls on the internet. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 81–91 (2005)
31. Wang, X., Reeves, D.S.: Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 20–29 (2003)
32. Wu, J., Wang, Y., Ding, L., Liao, X.: Improving performance of network covert timing channel through huffman coding. Mathematical and Computer Modelling 55(1), 69–79 (2012)
33. Zander, S., Armitage, G., Branch, P.: A survey of covert channels and countermeasures in computer network protocols. IEEE Communications Surveys & Tutorials 9(3), 44–57 (2007)
34. Zi, X., Yao, L., Pan, L., Li, J.: Implementing a passive network covert timing channel. Computers & Security 29(6), 686–696 (2010)