Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay

David L. Donoho[1], Ana Georgina Flesia[1],
Umesh Shankar[2], Vern Paxson[3],
Jason Coit[4], and Stuart Staniford[4]

[1] Department of Statistics, Stanford University
Sequoia Hall, 390 Serra Mall, Stanford, CA 94305-4065 USA
{donoho,flesia}@stanford.edu
[2] Department of Computer Science, University of California at Berkeley
567 Soda Hall, Berkeley, CA 94704
ushankar@cs.berkeley.edu
[3] International Computer Science Institute
1947 Center St. suite 600, Berkeley, CA 94704-1198
vern@icir.org
[4] Silicon Defense
203 F Street, suit E, Davis, CA95616, USA
{stuart,jasonc}@silicondefense.com

Abstract. Computer attackers frequently relay their attacks through a compromised host at an innocent site, thereby obscuring the true origin of the attack. There is a growing literature on ways to detect that an interactive connection into a site and another outbound from the site give evidence of such a "stepping stone." This has been done based on monitoring the access link connecting the site to the Internet (Eg. [7, 11, 8]). The earliest work was based on connection content comparisons but more recent work has relied on timing information in order to compare encrypted connections.

Past work on this problem has not yet attempted to cope with the ways in which intruders might attempt to modify their traffic to defeat stepping stone detection. In this paper we give the first consideration to constraining such intruder *evasion*. We present some unexpected results that show there are theoretical limits on the ability of attackers to disguise their traffic in this way for sufficiently long connections.

We consider evasions that consist of local jittering of packet arrival times (without addition and subtraction of packets), and also the addition of superfluous packets which will be removed later in the connection chain (chaff).

To counter such evasion, we assume that the intruder has a "maximum delay tolerance." By using wavelets and similar multiscale methods, we show that we can separate the short-term behavior of the streams – where the jittering or chaff indeed masks the correlation – from the long-term behavior of the streams – where the correlation remains.

It therefore appears, at least in principle, that there is an effective countermeasure to this particular evasion tactic, at least for sufficiently long-lived interactive connections.

Key Words and Phrases. Network intrusion detection. Evasion. Stepping Stone. Interactive Session. Multiscale Methods. Wavelets. Universal Keystroke Interarrival Distribution.

Acknowledgments. DLD and AGF would like to thank NSF ANI-008584 (ITR). AGF would like to thank the Statistics Department of UC Berkeley for its hospitality. JC and SS would like to thank DARPA contract N66001-00-C-8045.

1 Introduction

Perpetrators launching network intrusions over the Internet of course wish to evade surveillance. Of the many methods they use, one of the most common and effective is the construction of *stepping stones*. In this technique, the attacker uses a series of compromised hosts as relay machines and constructs a chain of interactive connections running on these hosts using protocols such as Telnet or SSH. The commands typed by the attacker on their own host are then passed along, unmodified, through the various hosts in the chain. The ultimate victim of the attack sees traffic coming from the final host in the chain, and because this is not the actual origin of the attack, little is revealed about the real location of the attacker.

An investigator seeking to locate the perpetrator would appear to be stymied by the need to execute a lengthy and administratively complex 'traceback' procedure, working back host by host, figuring out each predecessor in the chain step-by-step (based on whatever log records may be available at each stepping-stone site). For discussion of the use of stepping-stone attacks in high profile cases – and the difficulty of unraveling them – see for example [6] or [3].

An alternate paradigm for stepping-stone detection entails the installation of a *stepping-stone monitor* at the network access point of an organization (such as a university or other substantial local network). The monitor analyzes properties of both incoming and outgoing traffic looking for correlations between flows that would suggest the existence of a stepping stone [7, 11, 8]. See Figure 1.

This tradition of work has all been concerned with traceback of interactive connections: traceback of short non-interactive connections is harder and is presently unaddressed in the literature. Nor do we address it here. However, the interactive traceback problem is of interest, since there are many tasks that attackers must perform interactively. If the hacker has a goal beyond just compromising machines for zombies, if he or she really wishes to exploit a particular site for criminal ends, then the creative exploration and understanding of the newly compromised site requires a significant amount of human time, and for this one or more interactive sessions are highly desirable.

Attackers who are aware of the risk of monitors looking for stepping stones can attempt to evade detection of their stepping stones by modifying the streams

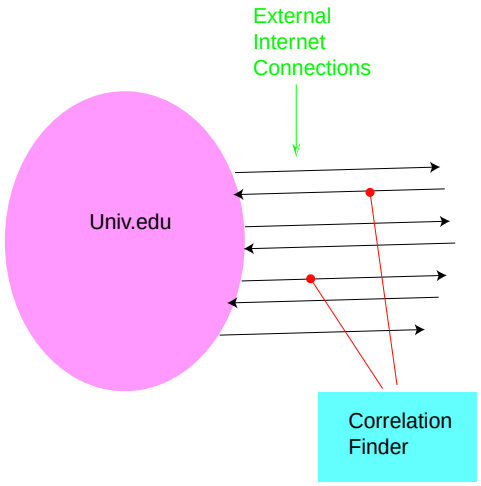


Fig. 1. Stepping-Stone Monitor

crossing the network access point so that they appear unrelated. Since the stepping-stone hosts are under their control, we must assume that attackers can arbitrarily modify their traffic in such evasion attempts. A wide spectrum of possible evasions might be considered; in the worst case, the information the attacker truly wishes to transmit could be embedded steganographically in connections that appear to be quite unrelated, both as regards content and traffic properties. On the other hand, such evasions might be very inconvenient to design, implement and use. It is of considerable interest to understand how well the various evasion techniques can work, and under what circumstances they can be defeated by monitoring techniques – particularly the ones that are easiest to deploy.

In this article we consider evasions based on keeping within the Telnet/SSH connection paradigm – which is obviously the most convenient for the attacker – and simply modifying the traffic being handled through the Telnet or SSH connections. We discuss a class of evasions to the monitoring techniques based on local timing perturbations, in which the stepping stone simply adds delay in more or less random amounts to traffic crossing the stepping stone. However, we assume that there is a maximum tolerable delay that the attacker is willing to introduce (since humans are not able to work effectively over interactive connections with very long latencies). We give a theoretical result that such *packet conserving* evasions are ineffective against appropriate correlation based on multiscale analysis using wavelets, at least in the limit of long connections.

We then consider the case of attackers who add extra packets into their connection, but still wish to have it look interactive. Again, we show that for long enough connections, it will be possible to correlate the two connections despite the added packets.

This suggests that the attacker wishing to evade stepping stone detection would be ill-advised to rely solely on local timing jitter or addition of chaff packets to connections. Based on our analysis, it appears that the most likely approach will require abandoning connection chains that use only standard interactive protocols, such as Telnet or SSH, for more sophisticated schemes, that can steganographically add traffic to connections that look like something else. These tools will be correspondingly harder to install and use. However, we also note that our results are primarily asymptotic, and require further analysis to determine the degree to which attackers can still evade detection within the framework by keeping their connections sufficiently short.

2 Previous Approaches

Staniford and Heberlein (1995) [7] initiated the literature of stepping-stone detection by considering chain-of-Telnet connections, in which the content is transmitted in the clear, and therefore could be statistically analyzed. Their approach was to tabulate character frequencies during set time intervals over all Telnet connections into and out of a domain, and to compare the tables of character frequencies looking for suspiciously good matches. As a technical feature, they used statistical analysis tools (principal components) to reduce the dimensionality of the feature vector, enabling rapid comparisons of features of different connections.

The increasing use of SSH and other encrypted modes of communication in recent years makes it important to develop a tool which does not require access to the actual transmitted content. Zhang and Paxson (2000) [8] developed an *activity*-based rather than *content*-based approach, which in particular could be used with chain-of-SSH connections. Their approach was based on the observation that interactive sessions have a strong "on–off" structure. Given this, one can then monitor the "off" periods of connections looking for suspicious coincidences of connections making nearly simultaneous transitions to an "on" period. They developed an on-line algorithm for looking for stepping stones and showed they could do this in practice at a large site (though it turned out that most of the stepping stones found were innocent).

Yoda and Etoh (2000) [11] also considered the problem of comparing interactive network connections that might be encrypted and they too relied on timing properties of the connection. However, they based their approach on looking at the average time lag between one connection and another, minimized over possible time offsets. They developed a quadratic time off-line algorithm for doing this comparison, and showed at least a preliminary ability to discriminate stepping stones pairs from unrelated connection pairs based on a threshold of about three seconds for the average delay between connections.

The mechanisms developed by Zhang/Paxson and Yoda/Etoh are both vulnerable to attackers perturbing the timing structure of their connections in order to defeat the stepping stone detection. In this paper we demonstrate that, for some types of perturbations, such vulnerabilities do not appear fundamental.

3 Next Generation Evasions

Existing approaches to detecting stepping-stones are subject to evasions, and the purpose of this paper is to analyze the capabilities of some of these evasions. The central issue is that attackers will have available the ability to effect *stream transformations* on the hosts in a stepping-stone chain, altering the relays from performing pure 'passthru' to instead modifying the stream in some way.

For example, in a Unix context, one could introduce filters for "chaff embedding" and "chaff stripping." Imagine a filter ⟨`enchaff`⟩ that merges the standard input with meaningless 'chaff' input from another source, so that the standard input content comprises only a sub-sequence of the output stream; ⟨`dechaff`⟩ extracts the embedded meaningful sub-sequence; and ⟨`passthru`⟩ simply copies standard input to standard output. By chaining together such filters, one can conceptually arrange that the content transmitted over the connection incoming to a site will not obey the sequencing and volume relationships of the outgoing connection, even though the semantic content is identical.

In fact, writing such filters in Unix is not quite trivial, due to buffering and pseudo-tty issues. But clearly they are realizable without a great deal of effort. Accordingly, we need to consider the possible impact of stream transformations used to evade stepping-stone detectors.

In short, the challenge for the next generation of stepping-stone monitors is to detect correlated activity between two streams when

- One or both streams may be transformed
- It is not possible to examine content for correlations

4 The Constraint Hypothesis

Our research began with the hypothesis that, while arbitrary stream transformations might conceivably be very effective at evading detections, certain constraints on interactive sessions might prevent the use of effective transformations.

For interactive connections, we argue for the following two constraints:

- *Latency constraints.* Ultimately, the chain of interactive connections is tied to a human user, for whom it will be annoying/tiring/error-prone to have to wait a long time for the results of their typing to be echoed or processed. Hence, we posit a *maximum tolerable delay* limiting what a stream transformation can impose; anything longer will be just too painful for the user.
- *Representative traffic.* Typing (and "think time" pauses) by humans manifests certain statistical regularities in the corresponding interpacket spacings, sharply distinct from machine-driven network communication. In particular, interpacket spacings above 200 msec (the majority) are well-described as reflecting a Pareto distribution with shape parameter $\alpha \approx 1.0$ [9]. A stream transformation which upsets this regularity can in principle call attention to itself and become itself a source of evident correlation between ingress and egress connections.

We can summarize these constraints as: *(i)* the original stream and its transformation must be synchronized to within a certain specific maximum tolerable delay, and *(ii)* the stream interarrival times must have the same distribution as the universal Pareto distribution described above.

This second constraint is particularly powerful. It seems difficult to add chaff to a stream without destroying invariant distributional properties, so in most of the remainder of this paper we consider schemes which do not add chaff. That is, we consider transforms that *conserve character counts*: each character in one stream corresponds to one character in the other stream. Such conservative transforms can only alter the interarrival times between items in the input stream and the output stream, and can be thought of as simply *jittering* the times to mask the similarity of the two streams.

We must, however, note an important caveat regarding constraint *(ii)*, which is that the Pareto distribution emerges as a general property when we analyze the statistics of many interactive interpacket times aggregated together. However, the *variation* in the distribution seen across different individual interactive sessions has not yet been characterized in the literature. It is possible that sufficient variation exists such that an attacker could inject chaff that significantly alters the distribution of the interpacket timings without an anomaly detector being able to flag the altered distribution as anomalous. With that possible limitation in mind, we now investigate what we can do to thwart evasion if in fact that the attacker cannot pursue such alterations. Later, we will return to the issue of detecting correlations despite the addition of chaff.

Assuming that the attacker is confined to conservative transforms, can they actually be used to hide the common source of two streams? To answer this question, we now examine possible evasion transforms that conform with the above assumptions.

One aproach is to use a transform which re-randomizes interarrival times. In words, we take a stream and 'strip it' of its identity by changing all the inter-keystroke times to a stochastically independent set of inter-keystroke times. Formally,

- Stream 1 contains Characters $c_1, ..., c_n$ at Times $t_1, ..., t_n$.
- Stream 2 contains the same Characters $c_1, ..., c_n$, at Times $u_1, ..., u_n$.
- The interarrival times $t_i - t_{i-1}$ are known to be independent and identically distributed (i.i.d.) according to a known distribution function F.
- Stream 2 is defined by interarrival times $(u_i - u_{i-1})$ which are also i.i.d. F, independently of (t_i).

This approach certainly removes all correlations between the two streams, but has two major flaws. First, it is not *causal*: it is possible that $u_i < t_i$ for certain characters i and $u_{i'} > t_{i'}$ for other characters i', while properly speaking, one of the streams must occur strictly after the other. (Which one occurs after the other depends on the monitor's location with respect to the position of the transformation element.)

It might appear that the difficulty with causality can be addressed by conceptually shifting the transformed stream in time by a fixed amount, preserving its

distribution but ensuring that we always have $u_i \geq t_i$. But a second problem remains: the two streams become unboundedly out-of-sync. Indeed, the difference between the two cumulative counting functions behaves as a random walk, and so we know from simple probability calculations that for this scheme, $|t_n - u_n|$ fluctuates unboundedly as $n \to \infty$, and that $Var\,(t_n - u_n) \geq constant \cdot n$ (essentially because $t_n - u_n$ is a sum of n i.i.d. random variables). It follows that for any given buffer length, eventually the delay between the two streams will surpass that length; and that for any tolerable perceptual delay, eventually the delay caused by the transcoding will exceed that length. In summary, transcoding to remove all correlations leads to complete desynchronization over time, violating the maximum tolerable delay constraint.

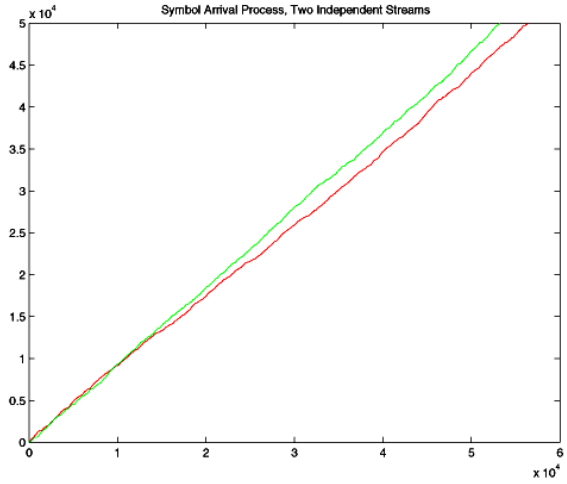


Fig. 2. Divergence of Independent Streams. The two cumulative counting functions diverge arbitrarily as time progresses.

The point is illustrated in the figure 2, which shows a simulation of the transformation discussed above, where keystroke arrivals are drawn from the empirical distribution given in [9]. We can see that after about 5,000 symbols, the two streams are about 500 characters out of sync.

How about partial randomization of keystroke arrival times? Consider the following local jittering algorithm, which we call *dyadic block reshuffling*. Given Stream 1 with arrival times t_i, this approach creates Stream 2 with arrival times u_i that never differ from those in Stream 1 by more than a certain specific guaranteed amount, but which are completely independent at fine levels. The approach has the following general structure:

– For dyadic intervals $[k2^j, (k+1)2^j)$, $N^1_{j,k}$ in Stream 1, compute arrival counts $N^1_{j,k}$.

- For a given 'scale' j_0, create Stream 2 so that $N^2_{j,k} = N^1_{j,k}$, for all $j \geq j_0$, all k.
- Method: identify all arrivals in $I_{j,k}$ and select random uniform arrivals in same interval.

The approach is illustrated in figures 3 and 4. The first one depicts the algorithm operating at a specific medium scale j_0: This sort of local shuffling does not suffer

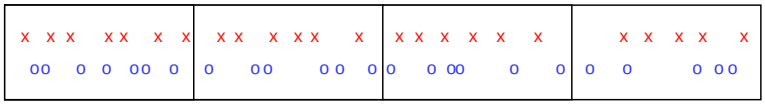


Fig. 3. Dyadic Block Reshuffling. Row of 'x': arrival times in original stream. Row of 'o': arrival times in transformed stream. Black Boxes: equi-spaced blocks of time. There are just as many times in each block for each stream. Times in transformed stream are chosen uniformly at random within block

from de-synchronization; as figure 4 shows, the two cumulative character counts functions cross regularly, at least once for each box.

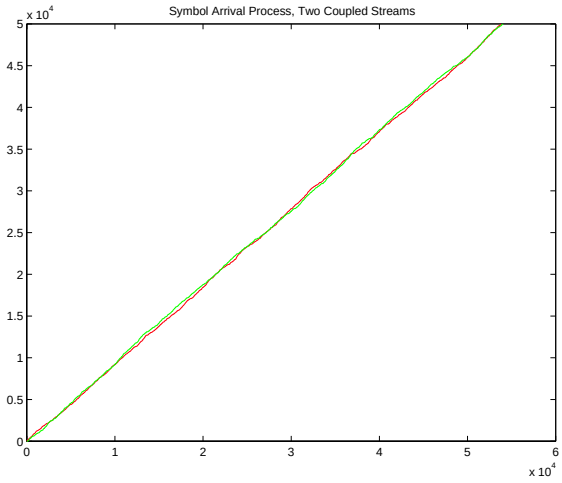


Fig. 4. Non-divergence of streams under dyadic block reshuffling

However, the constraint which is responsible for this crossing mechanism – $N^1_{j,k} = N^2_{j,k}$ at scales $j \geq j_0$ – says also that on sufficiently coarse scales the two counting functions agree, so there are measurable correlations between the two streams. We are confronted by a tradeoff:

- Pick j_0 at fine scale – we get tolerable delay but high correlation

– Pick j_0 at coarser scale – we get worse delay but reduced correlation.

Is this tradeoff inevitable? For example, are there local jitterings which are more cleverly constructed and which avoid such correlations?

We can formulate our central question in the following terms. Let $N_1(t)$ be the *cumulative character counting function* on the untransformed stream:

$$N_1(t) = \#\text{ of symbols in Stream 1 on } [0, t)$$

and similarly let $N_2(t)$ be the character counting function on the transformed stream. Our earlier discussion imposes specific constraints on these functions:

1. *Causality.* Characters cannot emerge from the transformed stream before they have emerged from the original stream. I.e., we must always have:

$$N_2(t) \leq N_1(t).$$

(The ordering of the inequality here is a convention; we could as well impose the reverse inequality, since, as discussed above, the direction of the causality between the two streams depends on the location of the monitor with respect to the transformation element.)

2. *Maximum Tolerable Delay.* Per the previous discussion, owing to human factors characters must emerge from the second stream within a time interval Δ after they emerged from the first stream:

$$N_2(t + \Delta) \geq N_1(t).$$

We then ask:

1. Do Causality & Maximum Tolerable Delay combine to imply noticeable correlations between properties of stream 1 and stream 2?
2. If so, what properties should we measure in order to observe such correlations?

5 Main result

Our principal result is a theoretical one, showing that *multiscale analysis of stream functions N_i will reveal, at sufficiently long time scales, substantial correlations.* To make this precise, we introduce a systematic multiscale machinery. Good references on multiscale analysis and wavelets abound, but we are particularly fond of [4, 5].

To begin, we fix a wavelet $\psi(t)$ which is either a 'bump' (like a bell curve) taking only positive values or a 'wiggle' taking both positive and negative values. See Figure 5 for some examples of each. We form a multiscale family of translates and dilates of ψ

$$\psi_{a,b} = \psi((t - b)/a)/a^p$$

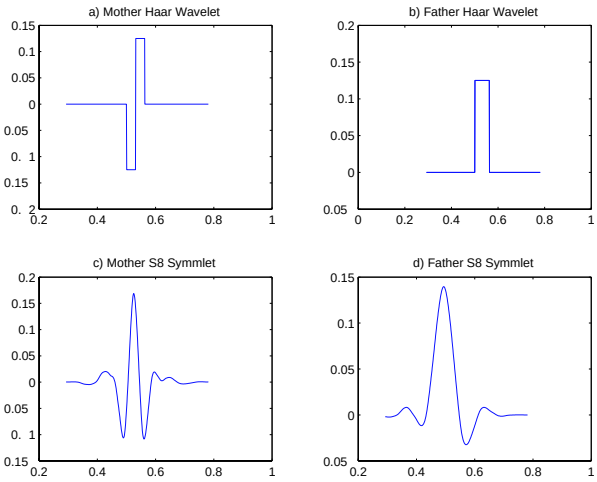


Fig. 5. Some wavelet waveforms: a) 'wiggle' (Mother Haar Wavelet); b) 'bump' (Father Haar wavelet); c) Mother S8 Symmlet; d) Father S8 Symmlet

Here the parameter p controls the kind of analysis we are doing. If ψ is a 'bump', we use $p = 1$; if ψ is a wiggle, we use $p = 1/2$. (The rationale for the different choices of p is given in the appendix.)

For computational reasons, we limit ourselves to a special collection of times and scales: the dyadic family $a = 2^j$, $b = k \cdot 2^j$. We can then use the fast wavelet transform to rapidly compute the *wavelet coefficients* of each stream function N_i, defined by

$$\alpha^i_{j,k} = \langle \psi_{a,b}, N_i \rangle$$

where $\langle f, g \rangle$ denotes the inner product

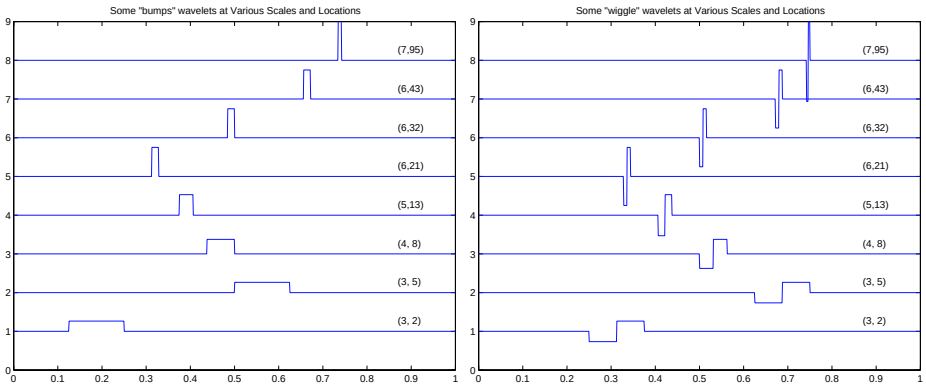
$$\langle f, g \rangle = \sum_t f(t) g(t).$$

When the wavelet is a 'bump', these are also called 'scaling coefficients'; when the wavelet is a 'wiggle' these are commonly called 'wavelet coefficients'. See Figure 6.

With this machinery, our central question becomes: if N_1 and N_2 obey the causality/maximum tolerable delay constraints, how similar are $\alpha^1_{j,k}$ and $\alpha^2_{j,k}$? In essence, we are analyzing the character counting functions of both streams across different time scales, looking for how similarly the character arrivals cluster at each time scale.

Our analysis follows two specific branches, depending on the choice of ψ.

– *Analysis by Multiscale Block Averages.* Here we choose ψ to be a very simple 'bump' – actually the "boxcar" function $\psi(t) = 1_{[0,1]}$ depicted in Figure 5,



(a) Father Wavelets (b) Mother Wavelets

Fig. 6. Scale/Location Families: 'bumps' and 'wiggles' at various scales and locations

panel (b). As indicated above, we choose $p = 1$, and it then turns out that the coefficients amount to simple averages of the data over blocks of various lengths and locations. Accordingly, we call this choice of ψ as corresponding to the analysis of "multiscale block averages."

We analyze the stream functions $N_i(t)$ via the dyadic boxcar family

$$\psi_{j,k}(t) = \psi((t - k2^j)/2^j)/2^j.$$

How similar are $\alpha^1_{j,k}$ and $\alpha^2_{j,k}$?

Our strategy for analysis is to estimate two quantities at each scale level j:

- The typical size of $\alpha^1_{j,k}$ at scale j; and
- The maximal deviation of $\alpha^1_{j,k} - \alpha^2_{j,k}$ at scale j.

We then compare these, and it will turn out that at long scales the deviation term is small compared to the typical size. Using analysis developed in the next section, we can then reach the following conclusions, explained here for the case of a Poisson input stream, where the analysis is simpler.

Suppose that $N_1(t)$ is a Poisson stream at rate λ. Then

- $\alpha^1_{j,k} \approx \lambda \pm O_P(1/\sqrt{scale})$, where $O_P()$ denotes the asymptotic order in probability.
- $|\alpha^1_{j,k} - \alpha^2_{j,k}| \leq O_P(\log(scale)/scale)$
- $|\alpha^1_{j,k} - \alpha^2_{j,k}| \ll |\alpha^1_{j,k}|$, at long time scales.

In words, the scaling coefficients of the two streams must be very similar at long time scales.

– *Multiscale Block Differences.* Here we choose ψ to be a very simple 'wiggle' – actually the Haar wavelet $\psi(t) = 1_{[1/2,1)} - 1_{[0,1/2)}$ depicted in Figure 5, panel (a); this is a simple difference of boxcars. As indicated above, we therefore choose $p = 1/2$, and it then turns out that the coefficients amount

to simple scaled differences of averages of the data over blocks of various lengths and locations. Accordingly, we call this choice of ψ as corresponding to the analysis of "multiscale block differences."

We analyze the stream functions $N_i(t)$ via the dyadic Haar family

$$\psi_{j,k}(t) = \psi((t - k2^j)/2^j)/2^{j/2}.$$

How similar are $\alpha^1_{j,k}$ and $\alpha^2_{j,k}$?

Our strategy for analysis is again to estimate two quantities:

- The typical size of $\alpha^1_{j,k}$ at level j; and
- The maximal deviation of $\alpha^1_{j,k} - \alpha^2_{j,k}$ at level j.

We then compare these and find that at long scales the deviation term is small compared to the typical size.

We reach the following conclusions, again in the case of a Poisson input stream.

Suppose that $N_1(t)$ is a Poisson stream at rate λ. Then

- $\alpha^1_{j,k} \approx O_P(1/\sqrt{scale})$.
- $|\alpha^1_{j,k} - \alpha^2_{j,k}| \leq O(\log(scale)/scale)$
- $|\alpha^1_{j,k} - \alpha^2_{j,k}| \ll |\alpha^1_{j,k}|$, at long time scales.

(The last two are identical to the results for the boxcar 'bump'; the first differs by the absence of the λ term.) In words: the wavelet coefficients of the two streams must be very similar at long scales.

This simple analytical result indicates, as we have said, that *character-conserving stream transformations which maintain causality and maximum tolerable delay, must also maintain correlations between streams at sufficiently long time scales.*

As stated so far, the result applies just to Poisson input streams. In the appendix we discuss extending the result to Pareto streams. This extension is of significant practical import, because the Pareto distribution, which as a model of network keystroke interarrivals is well supported by empirical data [9], is radically different in variability from the exponential distribution.

6 Analysis

In this section we develop some of the machinery used to support the results outlined in the previous section. Our first analytical tool for developing this result is a simple application of integration by parts. Let $\Psi = \Psi(t)$ be a function that is piecewise differentiable and which vanishes outside a finite interval. (This condition holds for both the Boxcar and the Haar wavelets). Then from integration by parts

$$\int \Psi \, \mathrm{d}N_1 - \int \Psi \, \mathrm{d}N_2 = \int \Psi \, \mathrm{d}(N_1 - N_2) = - \int (N_1 - N_2)(t) d\Psi(t)$$

so

$$\left| \int \Psi \, \mathrm{d}N_1 - \int \Psi \, \mathrm{d}N_2 \right| \leq TV(\Psi) \cdot \max\{|(N_1 - N_2)(t)| : t \in \mathrm{supp}(\Psi)\}$$

where $supp(\psi)$ – the support of ψ – the part of the t-axis where ψ is nonzero; and $TV(\Psi)$ – the Total Variation – is informally the sum of all the ups and downs in the graph of Ψ; formally, for a smooth function $\Psi(t)$, $TV(\Psi) = \int |\Psi'(t)|dt$, while for piecewise smooth functions, the total variation includes also the sum of the jumps across discontinuities.

Our second analytical tool has to do with properties of extreme values of stochastic processes. Causality and Maximum Tolerable Delay imply that

$$N_1(t) \geq N_2(t) \geq N_1(t - \Delta)$$

hence,

$$|N_1(t) - N_2(t)| \leq N_1(t) - N_1(t - \Delta)$$

and so

$$|N_1(t) - N_2(t)| \leq \max \{N_1(t + \Delta) - N_1(t) : t, t + \Delta \in \text{supp}(\Psi)\}.$$

In words, the difference between N_1 and N_2 is controlled by the volume in N_1. We now use results about extremes of Poisson processes. If N_1 is the set of cumulative arrivals of a Poisson counting process, then

$$\max \{N_1(t + \Delta) - N_1(t) : t, t + \Delta \in [a, b]\} \leq O_P(\log(b - a)) \cdot E\{N_1(t + \Delta) - N_1(t)\}$$

For more details see [2] and [1].

Based on these two analytical tools, we can easily obtain the results in the previous section:

- *Calculation for Multiscale Block Averages.* This is based on the following ingredients. First, symbols emerge at Poisson arrival times $t_1, \ldots, t_N$, with rate λ. Second, the 'bump' has mean 1 and so $E[\alpha^1_{j,k}] = \lambda$ (as one might guess). Third, $Var[\alpha^1_{j,k}] = Const \cdot \lambda/scale$, which is the usual $1/n$-law for variances of means. Consequently, the random fluctuations of the scaling coefficients obey

  $$\alpha^1_{j,k} \approx \lambda \pm c/\sqrt{scale}$$

  To calculate the maximum fluctuation of $\alpha^1_{j,k} - \alpha^2_{j,k}$, we observe that $TV(\psi_{j,k}) \leq 4/scale$ and $\sup\{N_1(t + \Delta) - N_1(t) : t \in [a, a + scale]\} = O_P(\log(scale))$, giving the key conclusion $|\alpha^1_{j,k} - \alpha^2_{j,k}| \leq O(\log(scale)/scale)$.
- *Calculation for Multiscale Block Differences.* Again we assume that symbols emerge at Poisson arrival times $t_1, \ldots, t_N$, with rate λ. Second, the 'wiggle' has mean 0 and so $E[\alpha^1_{j,k}] = 0$ (again as one might guess). Third, $Var[\alpha^1_{j,k}] = Const \cdot \lambda/scale$, which is the usual $1/n$-law for variances of means. Consequently, the random fluctuations of the wavelet coefficients obey

  $$\alpha^1_{j,k} \approx \pm c/\sqrt{scale}$$

  The calculation of the maximum fluctuation of $\alpha^1_{j,k} - \alpha^2_{j,k}$ is as for multiscale block averages above.

We again note that this analysis, as it stands, applies only to Poisson streams. Further analysis, given in the appendix, indicates that the same type of analysis can apply to Pareto streams and many others as well.

7 Simulation

To illustrate the above results, consider a simple transcoder: *local inter-keystroke shuffling* (LIS). This transcoder works as follows. We buffer symbols for M consecutive symbols or Δ milliseconds, whichever comes first. Suppose that the times of the symbol arrivals into the incoming stream buffer are t_1,..., t_m, so that necessarily $m \leq M$ and $t_m - t_1 < \Delta$. We then compute the interarrival times of the symbols in the buffer, $\delta_1 = t_2 - t_1$, $\delta_2 = t_3 - t_2$, ... , $\delta_{m-1} = t_m - t_{m-1}$.

Given the number $\delta_1, \ldots, \delta_{m-1}$, we perform a random shuffling of the times, obtaining $\varepsilon_1, \ldots, \varepsilon_{m-1}$. Then we define a second set of times by

$$u_1 = t_m,\ u_2 = u_1 + \varepsilon_1,\ \ldots\ u_i = u_{i-1} + \varepsilon_{i-1},\ \ldots,$$

and we output symbols in the second stream at times u_i (we ignore here the processing time required for calculating the u_i, which is surely trivial in this case). Figure 7 illustrates the type of transformation obtained by LIS.

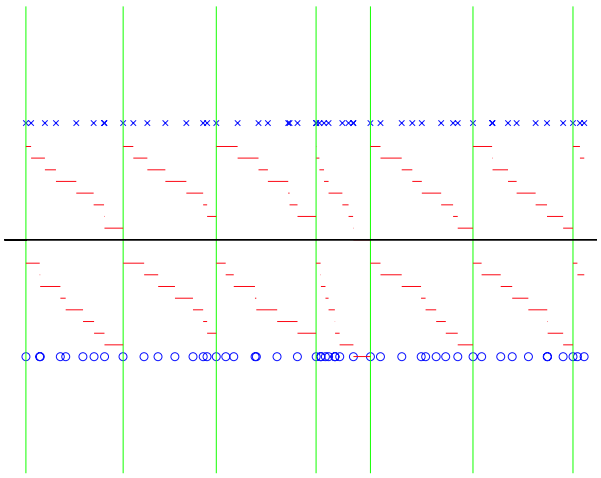


Fig. 7. LIS Transform: Row of 'x' – arrival times in original stream; Row of 'o' – arrival times in transformed stream; vertical lines demarcate zones of 8 characters; top group of horizontal lines – lengths depict inter-keystroke times; bottom group – lengths depict same times, but shuffled within one zone. *Note, the bottom set of boxes should be shifted in time over to the right by Δ; this will be fixed in the final version of the figure.*

A new stream synthesized by LIS has these properties:

- *Identical Distribution.* Whatever the underlying distribution of inter-keystroke times δ_i associated with (t_i), the new stream (u_i) has inter-keystroke times with the same distribution, because the actual inter-keystroke times are just the same numbers arriving in a different order.
- *Causality.* Characters arrive in Stream 2 later than Stream 1

$$t_i \leq u_i$$

- *Controlled delay.* Characters do not arrive much later in Stream 2:

$$t_i < u_i < t_i + 2\Delta.$$

Thus, there is no possibility that a statistical traffic anomaly detector can have cause for flagging a stream 2 produced by LIS as aberrant traffic. Also, by controlling the parameter Δ, we control the maximum tolerable delay.

To study the properties of multiscale detectors in the LIS setting, we use Monte-Carlo simulation. In our simulation experiments, we used for (t_i) samples from the empirical distribution of inter-keystroke times described in [9]. We created a stream several minutes in length and we transcoded this stream three times at three different delay parameters: 100 msec, 200 msec, and 300 msec.

From the streams, we created time series. We selected 256 second stretches of each stream, divided the time axis into 1/64th second intervals, and we counted the number of character arrivals (always 0 or 1, of course) within each interval. This gave us equally spaced series of 16384= 2^{15} 0's and 1's. We then used the freely available wavelet transform routines in WaveLab [10] to perform a wavelet analysis using the Haar wavelets.

The table below reports correlations between wavelet coefficients of the original stream and the three transformed streams. It contains the empirical correlations between the wavelet coefficients at various scales, defining the correlation coefficient at scale j by

$$Corr(j) = \sum_k \alpha^1_{j,k}\alpha^2_{j,k} / \left(\sum_k (\alpha^1_{j,k})^2 \cdot \sum_k (\alpha^2_{j,k})^2 \right)^{1/2}$$

where each sum is over all coefficients at a given scale j.

Scale	Δ =100 ms	200 ms	300 ms
32 sec	0.9964	0.9599	0.9382
64 sec	0.9965	0.9654	0.9371
128 sec	0.9966	0.9695	0.9458

These results are typical and repeatable. Correlations, of course, cannot exceed 1.0. So these correlations, which approach 1 at sufficiently long scales, are rather large. Evidently, given about 1 minute's worth of data on two jittered streams, we can obtain a substantial signal by correlation of wavelet coefficients. Note particularly the very high correlations when jittering is less than .1 sec.

8 Detecting Evasions in the Presence of Chaff

In the analysis since Section 4, we have assumed that any stream transformation being used to disguise correlations was conservative – that is, it exactly preserves the number and content of keystrokes, but perhaps alters their timing.

We now discuss the more general situation when this is not the case. A reasonable way to begin modeling the general situation is to say that we have two cumulative character counting functions N_1 and N_2, and that now

$$N_2(t) = N_1'(t) + M(t),$$

where $N_1'(t)$ is the cumulative counting function of the input character arrival times, perhaps after a conservative stream transformation, and M is the cumulative counting function of chaff arrival times. In short, as we watch characters coming in, some are from the input stream – only jittered – and others are chaff.

Suppose that we once again compute the statistic $Corr(j)$ at each scale. What happens? The results of course change. Suppose for simplicity that the chaff arrives according to the universal keyclick interarrival process, and that the chaff arrival process is stochastically independent of the N_1 process. Then one can show that instead of

$$Corr(j) \to 1, \quad j \to \infty,$$

we actually have

$$Corr(j) \to \rho, \quad j \to \infty,$$

where $0 < \rho < 1$. Here ρ may be interpreted as a 'signal/(signal+noise)' ratio, meaning that the limiting value ρ can be very small if the relative amount of chaff is very large, whereas it will be close to 1 if the fraction of chaff is negligible.

Nevertheless, no matter how small ρ might be, *any* nonzero value for ρ will be detectable, at least for sufficiently long-lived connections. Indeed, for large enough n, it will be clear that the empirical fluctuations in correlations due to statistical sampling effects are simply too small to cause observed values of $Corr(j)$ which are substantially nonzero. Given more space, we would provide a detailed analysis of this effect, showing that the mathematics predicts a substantial correlation between wavelet coefficients of N_1 and of N_2. The analysis is entirely parallel to the analysis given in earlier sections.

In short, although certainly the presence of chaff causes a more complex problem, the statistical tools and diagnostic approaches suggested for the no-chaff case seem to be equally applicable to the chaff case.

9 Discussion

This paper has considered basic 'proof of concept' issues. In particular, we have not discussed here the systems-level issues of working in a setting where there

may be many hundreds of active Telnet or SSH connections into and out of a large site (say, a university or corporate network), and it is required to monitor and detect stepping stones in real time. A typical issue would be to consider as a monitoring interval a specific length of time (e.g. 4 minutes), and calculate the proper height of the threshold for the 'stepping stone alarm' in order that in looking at thousands of pairs of connections which are truly uncorrelated, we control the false alarm rate to a tolerable number of false alarms per day. Obviously, it would be very important to study such issues carefully.

We have discussed here the fact that real interactive sessions seem to have inter-keystroke times whose distribution is Pareto in the upper tail. In the analysis section of this paper we have considered Poisson streams, which are easy to analyze. In the appendix, we show that the analysis can generalize to other streams. This points out that an accurate theoretical model for inter-keystroke timing is in order, so that we can focus attention and develop mathematical analysis associated with that model. Such a correct model would be extremely useful in practical terms, for example in systems-level work where it could used for false alarm calibration purposes.

Two particular components of the inter-keystroke timing model which should be considered more closely: (a) the correlation structure of adjacent/nearby inter-keystroke times; and (b) the chance of seeing many characters in a very short interval. The significance of these can be gleaned from the discussion in the appendix. Knowing more about either or both components would help mathematical analysis and simulation accuracy.

There are also other sources of information that we haven't discussed – the key one being the two-way nature of interactive sessions. There is far more information than just the keystrokes on the forward path through the stepping stones, there are also the echoes and command output on the reverse path, and it should be possible to use information about these to substantially improve detection.

References

1. Aldous, D.L.: Probability Approximations Via the Poisson Clumping Heuristic. Springer-Verlag, New York. January 1989
2. Lindgren, G., Leadbetter, M.R., and Rootzn, H.: Extremes and related properties of stationary sequences and processes. Springer, New York (1983). Russian translation; Nauka: Moscow (1988).
3. Shimomura, T. and Markoff, J.: Takedown. The pursuit and capture of Kevin Mitnick, America's most wanted computer outlaw–by the man who did it. Hyperion. December 1995.
4. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press. Second Edition, 2000.
5. Meyer, Y.:Wavelets: Algorithms and Applications. SIAM.May 1993
6. Stoll, C.: The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage. Pocket Books. October 2000

7. Staniford-Chen, S. and Heberlein, L.: Holding Intruders Accountable on the Internet. Proceedings of the 1995 IEEE Symposium on Security and Privacy, Oakland, CA (1995)
8. Zhang, Y. and Paxson, V.: Detecting stepping stones. Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, August 2000. *http://www.aciri.org/vern/papers/stepping-sec00.ps.gz*
9. Paxson, V. and Floyd, S.: Wide-Area Traffic: The Failure of Poisson Modeling. IEEE/ACM Transactions on Networking, Vol. 3(3),June 1995, 226–244
10. Wavelab Toolbox for Wavelet Analysis. Requires Matlab. *http://www-stat.stanford.edu/ wavelab*
11. Yoda, K. and Etoh, H.: Finding a Connection Chain for Tracing Intruders, In: Guppens, F.,Deswarte, Y., Gollamann, D. and Waidner, M. (eds): 6th European Symposium on Research in Computer Security - ESORICS 2000 LNCS -1985, Toulouse, France, Oct 2000

Appendix

Explanation of the Bumps/Wiggles Dichotomy

Analysis by "multiscale bumps" provides, as explained above, a collection of multiscale block averages. In other words, the coefficients measure the *rate* of typing of the given stream.

Analysis by "multiscale wiggles" provides, as explained above, a collection of multiscale differences of block averages. In other words, the coefficients measure the *changes* in the rate of typing of the given stream.

It is our opinion that measuring changes in rate, and noticing the times those occur, provides more reliable evidence for the identity of two streams; so we believe that analysis by multiscale wiggles (i.e. what is ordinarily called simply wavelet analysis) will give more reliable information indicating the identity of the two streams.

(There is one other advantage of wavelet analysis: the possibility of developing detectors for non-keystroke conserving schemes which work by the multiplexing of constant-rate chaff together with the original stream. Suppose that two streams differ in that stream 2 contains stream 1 along with characters from an independent chaff source of constant rate (e.g. Poisson with Rate 20 char/sec). It can be shown, by elaborating the point of view here, that the wavelet coefficients at sufficiently long scales will have a dependable correlation < 1, but which is stable and nonzero, and determined by a kind of statistical signal/chaff ratio. So we might notice that two streams which should be completely uncorrelated actually exhibit correlations which are definitely nonzero).

The different normalization of the wavelet coefficients in the two cases has to do with the appropriate means of interpretation of each type of coefficient. Averages are directly interpretable in the units of the phenomenon being measured, no matter what the scale of the average. Differences are not so universally interpretable; the convention $p = 1/2$ ensures that they are normalized according to the square-root of interval size rather than interval size. The rationale is that for typical point processes, the coefficients at different scales will then be of comparable size.

Generalization to non-Poisson Streams

The reader will note that only in two places in the argument of Section 6 did we use the Poisson process assumption

The first was

$$\max\{N_1(t+\Delta) - N_1(t) : t, t+\Delta \in [a,b]\} \leq O_P(\log(b-a)) \cdot E\{N_1(t+\Delta) - N_1(t)\}.$$

This condition says that *within any maximum tolerable delay interval, we are very unlikely to see character counts dramatically greater than the average character counts.* This inequality is extremely easy to satisfy and many point processes will obey it. It is also the case that real data will obey it. We might for example stipulate that no actual human person is ever going to exceed an absolute maximum of K characters in Δ no matter how long we wait. If we do, the above inequality will automatically be true, because $\log(b-a)$ grows unboundedly with observation period, while K is an absolute constant.

Incidentally, the Pareto nature of the upper half of the inter-keystroke timing distribution described in [9] is entirely compatible with this inequality. Indeed, the Pareto upper tail is responsible for occasional long dead spots in a stream, where no characters emerge. It is the lower tail – near zero inter-keystroke spacing – that determines whether the needed condition holds. The Poisson assumption makes the inter-keystroke distribution have a density $e^{-t/\lambda}/\lambda$ which is of course bounded near $t = 0$; this boundedness implies that there will not typically be large numbers of events in a short time. It seems safe to say that this aspect of the Poisson distribution accurately models real streams of keystrokes.

The second fact used was (in, say, the multiscale block averages case)

$$Var[N_1(0,T]] \asymp Const \cdot T$$

which says that the fluctuation in the number of events per unit time within an interval grows like the square root of the interval size. This will be true for many stationary point processes.

Now, the Pareto nature of the upper half of the inter-keystroke timing distribution described in [9], and the possibility of a non-i.i.d. behavior of inter-keystroke times can modify this inequality, even making the variability grow like a power T^β with $\beta \neq 1$. A more detailed analysis shows that even though the variance scaling exponents could be different, the fundamental behavior of the corresponding terms in the analysis would be the same.

Since our simulations indicate that the multiscale diagnostics work very well in the Pareto case, we omit further discussion of the mathematical details of the extension.