A Correlation Method for Establishing Provenance of Timestamps in Digital Evidence

By

Bradley Schatz, George Mohay, Andrew Clark

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2006 USA

Lafayette, IN (Aug 14th - 16th)

available at www.sciencedirect.com

SCIENCE DIRECT®

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

A correlation method for establishing provenance of timestamps in digital evidence

Bradley Schatz[*], *George Mohay*, *Andrew Clark*

Information Security Institute, Queensland University of Technology, GPO Box 2434, Brisbane 4001, Australia

Keywords:
Digital forensics
Digital evidence
Event correlation
Reverse engineering
Timestamp interpretation

ABSTRACT

Establishing the time at which a particular event happened is a fundamental concern when relating cause and effect in any forensic investigation. Reliance on computer generated timestamps for correlating events is complicated by uncertainty as to clock skew and drift, environmental factors such as location and local time zone offsets, as well as human factors such as clock tampering. Establishing that a particular computer's temporal behaviour was consistent during its operation remains a challenge. The contributions of this paper are both a description of assumptions commonly made regarding the behaviour of clocks in computers, and empirical results demonstrating that real world behaviour diverges from the idealised or assumed behaviour. We present an approach for inferring the temporal behaviour of a particular computer over a range of time by correlating commonly available local machine timestamps with another source of timestamps. We show that a general characterisation of the passage of time may be inferred from an analysis of commonly available browser records.

1. Introduction

The use of timestamps in digital investigations is pervasive. Timestamps are increasingly used to relate events which happen in the digital realm to each other and to events which happen in the physical realm, helping to establish cause and effect. A well-known difficulty with timestamps, however, is how to interpret and relate the timestamps generated by separate computer clocks when they are not synchronised (Stevens, 2004). Commonly observed differences in time occur from computer to computer both due to location specific time variations (such as time zones) and also due to the rate of drift of the hardware clocks in modern computers.

Current approaches to inferring the real world interpretation of timestamps assume idealised models of computer clock time, eschewing influences such as synchronisation and deliberate clock tampering. For example, in order to determine the clock skew of a computer being seized, it is commonly recommended that a record be made "of the CMOS time on seized or examined system units in relation to actual time, obtainable using radio signal clocks or via the Internet using reliable time servers."(Boyd and Forster, 2004). CERT recommend that, "As you collect a suspicious systems current date, time and command history ... determine if there is any discrepancy between the collected time and date and the actual time and date within your time zone" (Nolan et al., 2005).

While this approach will approximately identify the skew between the local time and the observed computer time at the time of the check, it says nothing about the passage of time on the computer's clock prior to that point (Weil, 2002). Uncertainty remains as to the behaviour of the clock of the suspect computer prior to seizure.

[*] *Corresponding author.*
E-mail addresses: b.schatz@qut.edu.au (B. Schatz), g.mohay@qut.edu.au (G. Mohay), a.clark@qut.edu.au (A. Clark).
1742-2876/$ – see front matter © 2006 DFRWS. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2006.06.009

In this paper, we explore two themes related to this uncertainty. Firstly, we investigate whether it is reasonable to assume uniform behaviour of computer clocks over time, and test these assumptions by attempting to characterise how computer clocks behave in the wild. Secondly, we investigate the feasibility of automatically identifying the local time on a computer by correlating timestamps embedded in digital evidence with corroborative time sources.

The paper is structured as follows. Section 2 presents an account of background material related to timekeeping and computer clocks. An experiment studying the behaviour of the local clocks in a computer network is presented in Section 3, together with an analysis of results. A second experiment attempting to identify a reliable means of inferring a suspect computer's timescale is then presented in Section 4. This experiment uses the results from Section 3 as baseline for comparison with the results generated by two correlation algorithms. Finally the two correlation algorithms are compared and conclusions drawn in Section 5 of the paper.

2. Background and related work

This section provides a brief introduction to the fundamentals of keeping time with computer clocks in our current era. The section concludes with a discussion of related work regarding the use of timestamps in forensics.

2.1. An introduction to computer timekeeping

A battery powered Real Time Clock (RTC) (also called BIOS or CMOS clock) is used to keep time while a computer is switched off. While the RTC is used as the basis for determining time when the computer boots, the interpretation of said time is operating system specific. For example, the family of Windows operating systems interprets the RTC as civil time, whereas the Linux operating system may interpret the RTC as either civil time or UTC by configuration.

Commonly, UNIX operating systems implement a software clock (called the *system clock*) by setting a counter from the RTC at boot, and employ a hardware timer (such as RTC timer interrupts, an APIC or other means) as an oscillator. Stevens suggests that all instances of the Windows OS base their timescale on the RTC throughout operation (Stevens, 2004). However, there is evidence to suggest that, similar to UNIX implementations, Windows 2000 and above similarly employ a software clock rather than using the RTC directly (Microsoft Corporation, 2006a,b; Sandhill Consulting, 1998).

2.2. Reliable time synchronisation

PC Clocks are commonly known to be inaccurate, due to the inherent instability of the crystal oscillators with which clocks are implemented. These vary widely due to temperature, voltage and noise (Mills, 2003). Since the late 1980s, significant effort has been invested in the development of techniques for obtaining reliable sources of time via directly connected atomic clocks, radio clocks, GPS and the Network Time Protocol (NTP). As far back as 1994, researchers were demonstrating synchronisation of computer clocks to an accuracy of 10 ms across the Pacific Ocean using NTP (Mills, 1994). Today, system clocks on the UNIX platform are able to be synchronised to reliable time sources on a nanosecond scale.

NTP is the most prevalent method of time synchronisation on UNIX hosts. From Windows 2000 onwards, Microsoft has included in their operating systems a restricted version of NTP, called Simple Network Time Protocol (SNTP). While this is compatible with NTP, SNTP does not implement the clock discipline algorithms present in the former, and is not capable of delivering the same degree of precision.

The degree to which reliable network time may be utilized by a particular PC running Windows varies. By default, Windows 2000 workstations have the SNTP service switched off, while Windows XP workstations by default will synchronise with an SNTP service hosted at time.windows.com once every week. Both 2000 and XP workstations in a domain network will however synchronise via SNTP from the domain controller.

In theory then, stand alone Windows XP workstations will become synchronised with civil time by use of SNTP once a week. Stand alone Windows 2000 PC's will likely drift away from civil time.

2.3. Factors affecting timekeeping accuracy

A number of interrelated factors influence the accuracy of both timekeeping on computers, and the interpretation of timestamps sourced from them. We summarise these below:

System clock implementation: as discussed previously, the quartz crystals used as oscillators in computers are notoriously unstable. Implementing the correct local time offsets for civil time is complicated by changes in region-specific time zones. Recent evidence for the importance of this may be illustrated by the flurry of patches related to the Melbourne Commonwealth Games daylight savings time extension in early 2006 (Microsoft Corporation, 2006c).

Clock configuration: it is common to see Windows workstations with the time zone set to the default installation time zone. Another clock configuration error is the commonly occurring example of systems where the BIOS time has not been correctly set.

Tampering: the practise of setting computer clocks back or forward for reasons such as evading digital rights management or misdirection of investigation is often referred to as tampering. Timestamps, like any data, are subject to the possibility of deliberate modification.

Synchronisation protocol: the Windows time synchronisation protocol is based on SNTP and is only designed to keep computers synchronised to within 2 s in a particular site and 20 s within a distributed enterprise. Computers using NTP and SNTP without cryptographic authentication are subject to protocol-based attacks.

Misinterpretation: timestamps are related to a particular frame of reference, and their correct interpretation requires knowledge of that context. For example, in order to interpret at what time an Internet Explorer timestamp corresponds to in the civil time where it was generated, one needs to know the time zone offset. Other sources of uncertainty are the ambiguity as to what point in time the timestamp refers to for a particular event – is it the start time or the end time of the event – and was the timestamp generated at the time of the event or the time of writing to the event log.

Bugs: software errors in the implementation of software clocks or the timestamp serialisation algorithm have the potential for adversely affecting timekeeping accuracy.

2.4. *Usage of timestamps in forensics*

Aside from the comprehensive study of computer time behaviour on UNIX systems done in the context of developing the NTP infrastructure (Mills, 2003, 1994), to the best of our knowledge there has been little research in either characterising the behaviour of the timescale of unsynchronised Windows computers, or on automated means of identifying the timescale.

In the computer forensics literature, timelining is often referenced as an important tool in determining cause and effect. Stevens (2004) has proposed a model and algorithm for relating timestamps taken from multiple timelines. In this model, a base clock is set to UTC, and subordinate clocks are defined in terms of skews from parent clocks with additional skews further generated from time drift rates.

Gladyshev and Patel (2005) propose using corroborating sources of time to find the bounds of events with uncertain time, defining both a formalism and an algorithm for relating events with uncertain times. Their approach differs from ours in that we deal predominately with certain times, but uncertain skew.

Weil (2002) argues for dynamic analysis of the temporal behaviour of suspect systems, proposing correlation of timestamps embedded within locally cached web pages with the modified and accessed times (MAC times) of the cached files.

2.5. *Relation to existing work*

Our work has similar objectives but differs significantly from Weil in two respects. Firstly, we investigate to what degree timescales are unstable. Secondly, Weil's approach relies on manual classification of cached web pages as dynamically or statically generated. This is due to the technique relying specifically on dynamic content in order for the embedded timestamps to be interpreted. In addition, we also present two algorithms, which enable the automatic determination of the behaviour of a suspect computer's clock by comparison with a commonly logged corroborative source.

3. Characterising the behaviour of drifting clocks

Having enumerated the main factors influencing the temporal behaviour of the clock of a computer, we attempt here to experimentally validate whether one can make informed assumptions about such behaviour. We do this by empirically studying the temporal behaviour of a network of computers found in the wild.

The subject of our case study is a network of machines in active use by a small business. The network consists of a Windows 2000 domain, consisting of one Windows 2000 server, and a mixed number of Windows XP and 2000 workstations. Access to the Internet is provided by a Linux based firewall. In this case, the Windows 2000 domain controller (DC) (the server) has not been configured to synchronise with any reliable time source, and as such has been drifting away from the civil timescale for some time. The Linux firewall also provides both an squid[1] web proxy server, and an NTP server, which is synchronised with a stratum 2 NTP server[2] All workstations are configured to use the squid proxy cache for web access.

Our goal here is to observe both the temporal behaviour of the DC, and the effects of synchronisation on the subordinate workstation computers. We would expect that the timescales of the workstation computers would approximate that of the DC, due to the use of SNTP in this network arrangement (see Section 2.3).

In order to observe this behaviour, we have constructed a simple service that logs both the system time of a host computer and the civil time for the location, which we obtain via SNTP from the local NTP server. The program samples both sources of time and logs the results to a file.

The logging program was deployed on all workstations and the server on the 1st February 2006, and the results checked mid March. Unfortunately, the program was rendered short lived, as a particular bug in the Windows service implementation of Python (the implementation language) saw the log service crash after writing 4K of debug messages to the standard output steam. On fixing the bug, a new version was redeployed in the 21st March 2006 for 20 days, and then results collected.

3.1. *Discussion of results*

The graphs presented below are based on the sampled timescales taken from machines in the subject network. The *x*-axis is the time and date of the sample, taken from the civil timescale, as served by the NTP server. The *y*-axis is the difference in time between the system time and civil time at that moment, in seconds.

Fig. 1 is the graph of results taken from the domain controller of the Windows 2000 server based network. The solid line of samples shows a uniform drift of the system time away from civil time for the time period 21st March through 10th April. The other two sets of samples from the 1st February through the 21st March 2006 are samples taken by the initial version of the program in the time after a boot (before the program crashed). Extrapolating the solid line shows the drift of the server to be at a near uniform rate.

Fig. 2 shows results taken from a Windows 2000 workstation called Florence over the same period. It displays a general time drift trend which correlates[3] with the domain controller. The faulty logging service has generated far more samples than were generated in the case of the server. This is due to the habitual shutting down of the computer by the user at the end of each work day, resulting in a number of samples

[1] http://www.squid-cache.org/.

[2] Stratum refers to the distance from a reference clock.

[3] In this paper we use the word correlate to indicate a correspondence between two measured properties. We do not subscribe to the statistical meaning of the word.

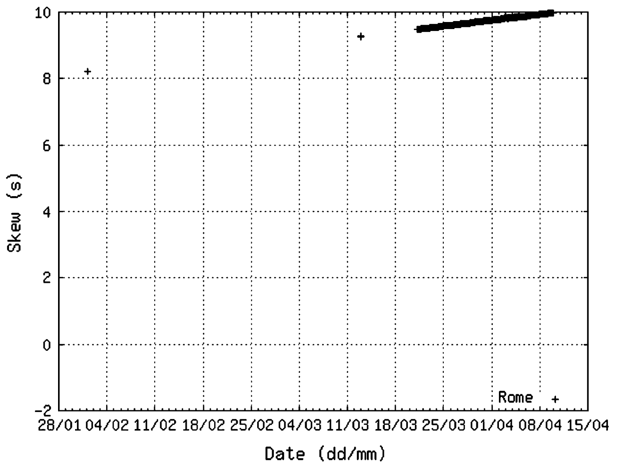

Fig. 1 – Clock skew of domain controller "Rome" offset from civil time.

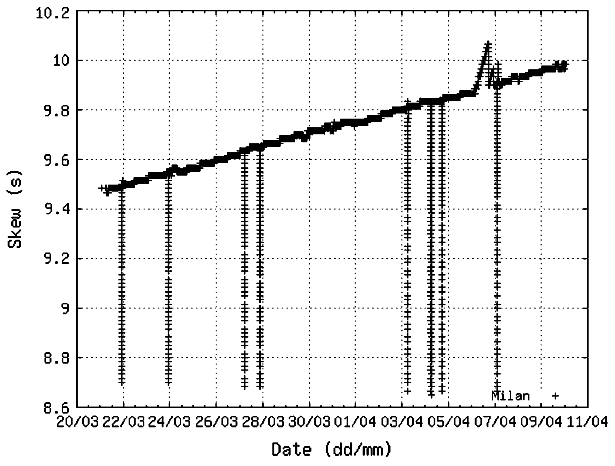

Fig. 3 – Clock skew of workstation "Milan" offset from civil time (zoomed).

generated every time the machine reboots (before the initial version of the program crashes).

The scale of the graph is misleading as to the number of outlier values present from 8:19:34 am through 8:25:56 am on the 20th February. The cross at 0 skew actually represents 38 outlier values, which do not fit a model of time where the clock is synchronised to the DC. It seems highly irregular that during this period the machine became synchronised to within one second of civil time (a time stream which the network in question has no configured reference to).

The default auditing configuration of the Windows network did not include the necessary privilege to identify whether this was user instigated. However, the accuracy to which the clock became synchronised with civil time leads us to suspect that this was not the result of user interaction; rather that it was the action of some program which had access to an external, reliable time source. The Windows update service was active during this period; however, we have no evidence to support the theory that this was the cause of synchronisation with the civil timescale during this period.

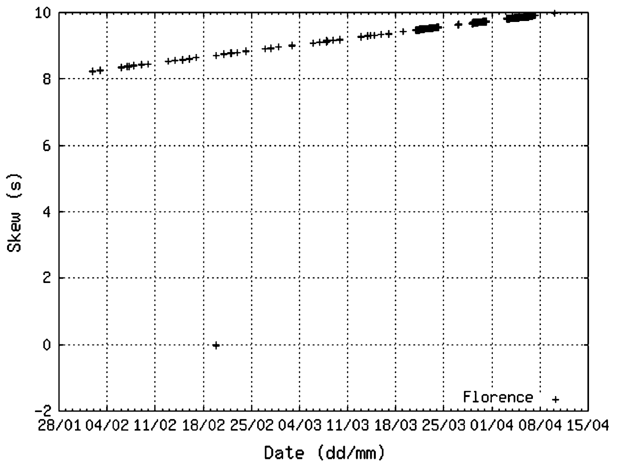

Fig. 2 – Clock skew of workstation "Florence" offset from civil time.

The graph presented in Fig. 3 shows the skew data taken from a Windows XP workstation named Milan.[4] Again the drift rate generally remains constant, and correlated with that of the server; however, there are two sets of anomalies which deviate from this general trend. Immediately noticeable are the almost vertical lines which indicate a resynchronisation with the DC timescale from wide time skews. Rebooting the computer, the second anomaly is the two peaks on the graph around the 6th and 7th April.

On closer investigation, the vertical line on the 3rd of March reveals that over a period of 44 min and 50 s of real time, the system clock only advanced 43 min 42 s. In total, the system clock loses 1 min 8 s over this period. Clearly, any timestamps generated during this period will be unreliable.

A recent check of this workstation's RTC via the BIOS configuration interface revealed that RTC was minutes ahead of the system time measured just moments before. It would appear from this that either Windows XP does not update the RTC, or that update of this particular RTC failed. Interestingly, we see similar behaviour for the PC named "Trieste" (shown below in Fig. 4) which is the only other Windows XP host on the network. All four other workstations (which are running Windows 2000) do not exhibit this behaviour. The near linear relationship of the lower ends of the vertical reboot lines may indicate the rough drift of the RTC.[5]

The graph labelled Fig. 5 represents a combination of data from the DC and Milan about the period where peaks are seen in the skew graph. We can see here that the DC was maintaining a stable timescale (part of its data is showing through under the peak) for the period while Milan began drifting away

[4] The scale of this graph differs from the previous graphs in order to present a more clear view of the features in discussion. We note that the overall form of the graph when taken at the previous scale follows the same gradient and offset.

[5] As the focus of this section is on describing observed deviations of Windows based clocks from the ideal, we leave experiments which conclusively determine the behaviour of the Windows XP clock at boot to others.

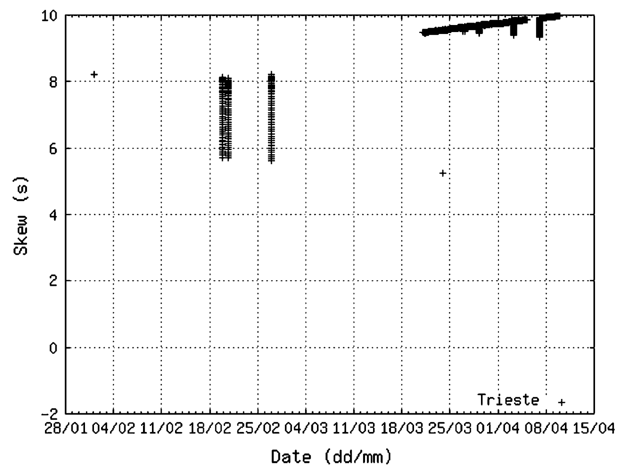

Fig. 4 – Clock skew of workstation "Trieste" offset from civil time.

from the timescale of the DC. At the start of the peak we can see that Milan began drifting away from the DC at a rate of around 1 s every 14 min, before re-synchronising with the DC.

Investigation of the event logs of the computer revealed an inordinate number of Print subsystem warning events in the system logs (which appeared to indicate repeated retries of installation of a print driver) before this time. No other events of interest were found. This drift is unlikely to have been based on a single operator action, as the corresponding change in skew would have been immediately visible, with a discontinuity between the two points.

The remaining three workstations stayed synchronised with the DC, with no temporal anomalies observed. The skew timelines for these were similar to Fig. 1. For reasons of brevity they are not reproduced here.

From these results we make a number of conclusions.

In general, we find that Windows hosts (2K and XP) integrated with a Windows based time synchronisation network will stay synchronised. However, the anomalies observed above indicate that making reliable statements about the timescale of a particular workstation computer within a Windows domain network (and as such the interpretation of timestamps from these workstations) is problematic.

Windows computers, not in a domain network, either tethered to a reliable source of time, or loosely tethered (such as computers running the XP OS) may suffer from the same problem. The observation that the host "Milan" became synchronised with civil time for a period, and the further observation of it drifting away from the DC timescale and civil time (for no observable reasons) indicate that other factors are influencing the behaviour of the clock.

4. Identifying computer timescales by correlation with corroborating sources

Given our uncertainty with respect to the timescale of a particular computer, we seek automated methods for identifying the temporal behaviour. In this section, we describe an automated approach which correlates timestamped events found on a suspect computer with timestamped events from a more reliable, corroborating source.

Web browser records are increasingly employed as evidence in investigations, and are a rich source of timestamped data. The corresponding ISP side records which correspond to these are proxy logs. We expect that the common practise of deploying transparent proxies by ISPs will see these logs also increasingly used.

Due to the increasing ubiquity of web browsers, we have chosen to use the web browser and proxy records as data sources for use in identifying temporal behaviour. We expect here that in the process of an investigation, proxy logs which relate to a suspect computer may be obtained from the ISP which has served as the computer's gateway to the Internet.

We assume that these records on the proxy would be produced by a computer which is synchronised with an accurate time source. (While this might not currently be a safe assumption, we look towards a near future where the provenance of audit records receives closer attention.)

The web browser records are from the target computer, of uncertain behaviour which is to be clarified. For this particular experiment we consider the records generated by the Internet Explorer (IE) web browser prevalent on Microsoft operating systems, along with the proxy logs created by the squid proxy cache.

IE stores records of browsing access in two subsystems: the cache and history. These records are all stored in separate files, all called *index.dat*, but located in different directories.

The IE cache subsystem stores locally cached copies of web content, such as pages and images. An index is kept mapping web addresses to these locally stored copies in a file called *index.dat*. The cache index files contain entries for all cacheable resources visited, including component files of a particular viewable page (for example, images, sounds, flash animations, etc).

The history subsystem creates a historical record of pages visited over time in a set of index.dat files. Three separate types of history files are kept: the root history, daily sort

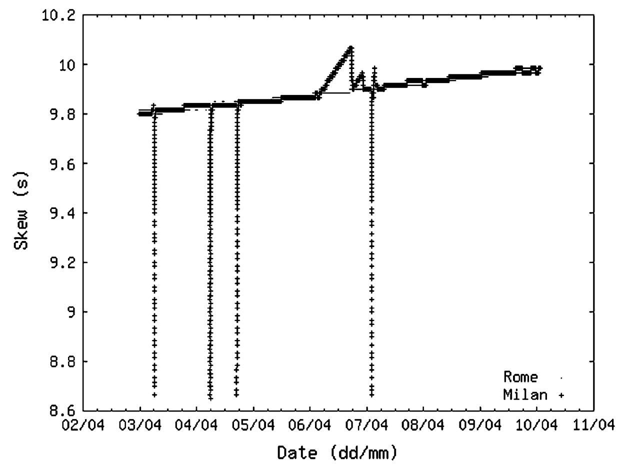

Fig. 5 – Clock skew of "Rome" vs. "Milan" offset from civil time (zoomed).

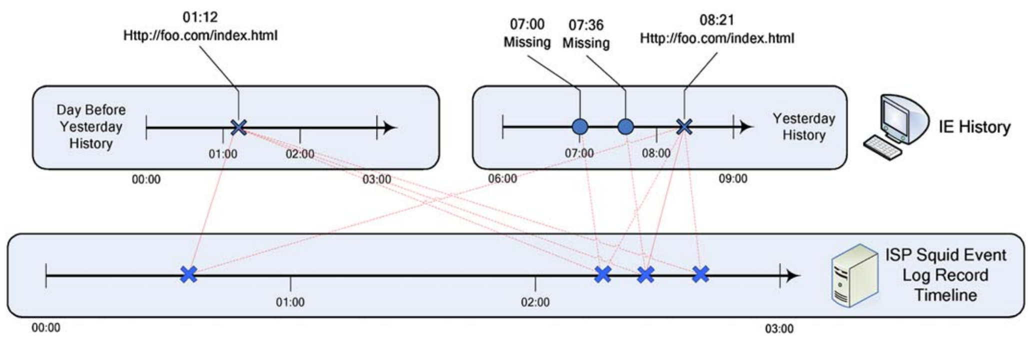

Fig. 6 – Matching is complicated by only the most recent record present in the history.

history and weekly sort history. Within these files are records of visits to top-level viewable pages:

- Pages visited by typing a URL
- Pages visited by clicking on a hypertext link
- Documents opened within Windows explorer clicking (i.e. .xls, .doc,...)

The cache and history *index.dat* files are all of a similar undocumented binary file format. Despite the lack of documentation, there exist a number of documented analyses of reverse engineering the file format, and a number of tools are available which will interpret the content of this file. For a good description of the file format, especially notable in distinguishing some subtle semantic differences in interpreting the timestamps in these records see Boyd and Forster, 2004.

We initially used the Pasco (Jones, 2004) tool for extracting the data contained in these files. We chose this tool due to its freely available source code. In practise, we suspected that it was generating spurious results. This prompted us to perform our own reverse engineering effort. Our new tool identified a bug in the Pasco tool where a spurious record was generated due to an unchecked file read for an offset outside the bounds of the file.[6]

The squid proxy cache logs a record of all web transactions which it processes in a file called *access.log*. This is a textual log file. The fields of interest to us are the resource access time (which, similar to the IE index files is the end of the transaction), and the URL visited.

4.1. *Challenges in correlating these sources*

Our experiment involved translating the web browser records and squid logs into a common representation and matching entries from both sources based on the URL visited. We assume that the last accessed time from the squid record is in fact civil time (kept tightly synchronised using, for example, NTP), and compare that time with the last accessed time from the corresponding history or cache record.

The primary challenge related to correlation is in determining which entry in the squid cache log corresponds to a particular entry in the cache or history records. As IE records are most recently used records (MRU), there will not be a one to one mapping between history entries and squid events. We illustrate this with the following example.

Fig. 6 depicts the relationship between records of visits to a particular page over two days. On the first day, the user has visited the site once, and on the second day has visited the site a further three times. However, as the history is a MRU record, the visits at 7:00 and 7:36 are effectively forgotten from the IE history, with the visit at 8:21 being the only record left for that day. Simple matching based on the URL field of each record will result in four potential matches for each history file record. The addition of further history records and cache records related to visits to this URL complicates the matter even further. Our correlation approach must, in this case, determine which potential match is the correct match.

4.2. *Evaluation methodology*

For the two algorithms explored, the sampled timescales from the previous experiment in Section 3 are used as a baseline for determining which matches are true or false. True positives are data points' output by the correlation algorithm which correlate with the timescale identified in Section 3. False positives are matches generated by the algorithm which do not correlate with the timescale. True negatives are prospective matches that are rightly discarded by the algorithm, while false negatives are data points which would correlate with the timescale, but the algorithm classifies or misidentifies as not correlated.

4.3. *Clickstream correlation algorithm*

Our initial approach to correlation is based on the concept of *clickstream*. We borrow this term from the web content industry, where it refers to the path taken by an individual visitor navigating a website in a particular session. Our hypothesis was that for a particular clickstream, the time periods between hits (visits to a particular URL) would form a unique signature.

We define a clickstream as a time ordered sequence of page hits within a website. The *intra-hit time* is the time period between two successive page hit events in a clickstream. We limit the definition of clickstream such that the intra-hit time for successive hits is within *max* seconds of each other and further than *min* seconds apart. We define a maximum limit so that we may disambiguate sessions. The function of the minimum is described further below. Finally, the *dimensions* of a clickstream are the ordered set of intra-hit times.

[6] Our new parsing tool, imaginatively named pasco2, is available at http://www.bschatz.org/2006/pasco2/.

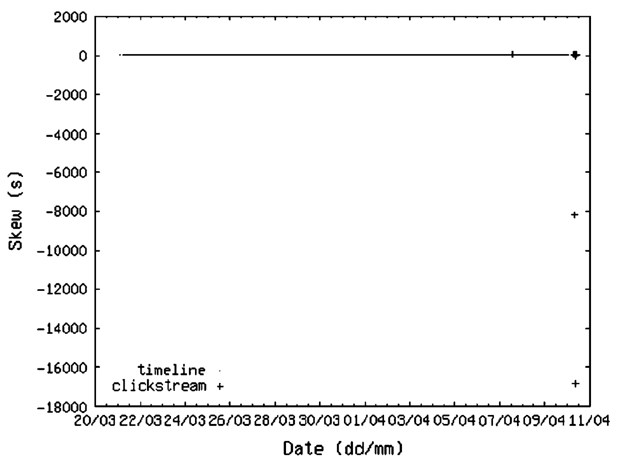

Fig. 7 – Correlated skew vs. experimental skew for host "Milan" do not correlate due to the presence of false positives.

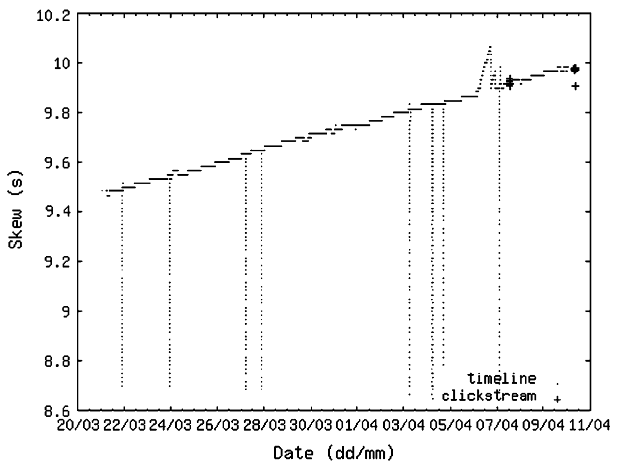

Fig. 8 – Correlated skew vs. experimental skew for host "Milan" correlates when false positives are removed.

The algorithm attempts to fit clickstreams identified in the web browser records to the squid event stream to corresponding events in the squid logs. The heuristic here is that the longer the clickstream, the more unique the dimensions of it will be, thus giving a single unique match when fitting to the other event stream.

Figs. 7 and 8 are of a clickstream correlation run, graphed with the timescale log of the workstation "Milan". The clickstream correlation dataset in Fig. 7 contains 75 results, of which we can see four clusters of results.[7] Clearly there are conflicting data. The two clusters are visible but not on the timeline actually contain five false positive values which are causing the problem. Removing these false positives from the result set results in the graph labelled Fig. 8, where we can see tight correlation with the workstation's timescale.

The results of running the same correlation algorithm on the host "Pompeii" which has generated far less web traffic over the period is presented in Fig. 9. In this case the clickstream correlation algorithm produces no false positives.

4.3.1. *Results*

In practise the rate of false positives increased when comparing intra-hit times at magnitudes below the magnitude of one second. We expect that this is due to the measurement error being more pronounced the smaller the intra-hit time becomes. Values of around 20 min for the maximum intra-hit time and values of over 1 s for the minimum value, produced clickstreams with the best uniqueness properties (as measured by reduction in rate of false positives).

Modifying the algorithm to filter clickstream acceptance based on clickstream length produced a similar effect on false positive rate, and consequently a high rate of false negatives for clickstreams of larger size. With larger sized clickstreams the rate of true positives falls off quickly however, and the rate of false negatives becomes high.

The algorithm performed far better on cache records than on history records. We expect that this is due to the difference in granularity of record keeping in the sources. As the cache stores cache records both for top level web pages and component content such as images, style sheets and the like, clickstreams are more likely to be formed. The IE history subsystem only records the top level page views, so is less likely to produce long clickstreams in situations where users do not heavily explore websites.

Designing an algorithm which eliminates these false positives is complicated by the fact that the last access timestamp of any particular cache record is unreliable, as it may have been accessed more recently by the user (before the cached content expired). In this case, no corresponding squid event would be logged, thus introducing a skew to the expected offset of the matching squid event.

For this reason, we set about identifying a means of identifying IE records which must have been requested via the squid proxy and not from the local cache.

4.4. *Non-cached records correlation algorithm*

After some further investigation, documentation of another effort at reverse engineering the index.dat format came to

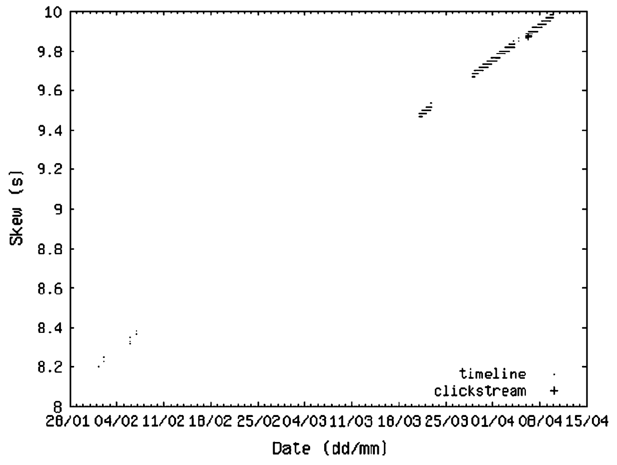

Fig. 9 – "Pompeii" cache correlation.

[7] We note here that a colour graph would be more illuminating, as the timeline values on the graph dominate.

light (Thomas, 2003). This work identified another field in the IE History record which recorded the total number of accesses to a particular web resource. For records where this field has a value of one, we can be sure that there has only been one access and that the record has come directly via the squid proxy.

Our new algorithm reduces uncertainty by choosing only history records which must have come directly via squid, bypassing the local cache. Furthermore it places a high value on matching entries for which there exists only one corresponding match in the squid log.

The algorithm is defined as follows. All potential matches of non-cached history records are found. A subset of these, called the *basis set* is identified, where for each history record, only one corresponding squid match is found. We call the set of remaining potential matches the *remainder set*. The *basis set mean* is the mean of the skews of each match in the basis set.

We further cluster the remainder set based on the history record common to potential matches. For each of these clusters, we find the potential match with a skew closest to the basis set mean, and add it to the initially empty *inferred set*, discarding the rest. The results of this algorithm are the union of the basis set and the inferred set.

4.4.1. *Results*

In practise this algorithm produces a set of data which correlates well with the timescales produced by our previous experiment. For example, Fig. 10 is a graph of the output of the algorithm described above overlaid over the timescale for host Milan obtained from the previous experiment.

Of 1188 unique history records, 821 potential matches were identified. One would expect that the number of potential matches would be higher, however URLs with encoded GET requests are not matched due to squid's anonymised logging of this type of URL.

In practise there are a significantly high proportion of non-cached hits in the history for our algorithm to work effectively. The algorithm identifies 304 potential non-cached matches, and a basis set of 110 matches from this. In total the algorithm generates 134 data points.

Comparison with the sampled timescale reveals numerous false positives (around 15–20) for which we have no explanation. On close examination, a number of data points were

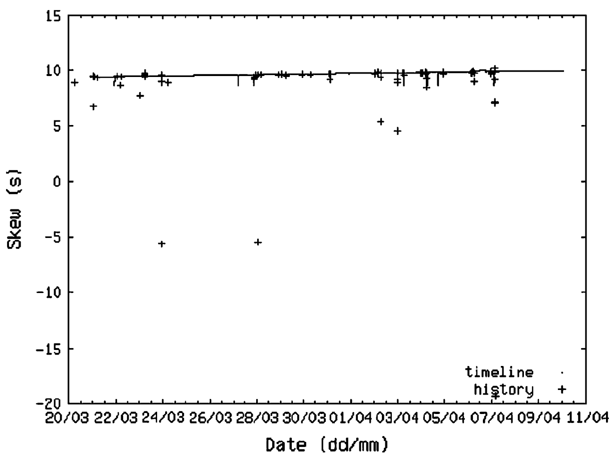

Fig. 10 – History correlation vs. timescale.

identified which correlated with the vertical sections immediately after reboots.

We are confident that the algorithm generates many false negatives due to its simplicity in selecting non-cached hits for the basis set. A more comprehensive algorithm would in addition to finding history records with an accessed count of 1, use the temporal ordering relationship between the history record sets. For example, say the oldest weekly sort file contains a record for a particular URL with accessed count equal to one. If a newer sort file contains a record for the same URL visit, with an accessed count of two then one can be sure that this record corresponds to a non-cached access.

4.5. *Discussion*

In this section the two algorithms are compared, and the general problems related to correlating these types of event logs are outlined.

Of the two algorithms, the history correlation algorithm performed the best. Results are generated which cover a far wider period of time than the cache oriented algorithm, giving greater insights into the temporal behaviour of the computer. Furthermore the ratio of true to false positives is far higher.

The history algorithm was originally the worst performing of the two approaches. At that point in time, determining whether the high rate of false positives was due to a tool implementation error or an error in the correlation algorithm was problematic. Boyd's paper (Boyd and Forster, 2004) was at that point in time essential in identifying that our interpretation of the weekly history timestamps was mistaken.

Despite having re-implemented a new set of *index.dat* file parsers (and discovered a third timestamp in the history records[8]) we still used the semantics defined by the pasco tool. Our model was corrected to treat the first timestamp in the weekly sort history record as the accessed time, offset by the local time zone offset in operation. This resulted in the high rate of positives and a low rate of false positives previously seen in Fig. 10.

Both approaches to developing a correlation algorithm outlined above make a closed world assumption – that the algorithm has access to all of the information that it needs. In practise, development of the algorithm was complicated by this not being the case. Consider, for example, Fig. 11. This graph was generated using the same history correlation algorithm as that seen in Fig. 10. Strong correlation with the computer's timescale is evident; however, there are in this case false positives in the extremes of the graph.

Examination of the false positives indicated that they were related to records from a particular squid access log. This access log had not been included in the correlation run for processing speed reasons. The omission of the records resulted in the algorithm picking a match from another squid

[8] A 32bit MSDOS timestamp was identified at offset 0×50 h within the history record. Within the root history file, interpretation of this timestamp is the last accessed time, as is apparent by comparing the last access time in the Internet Explorer history viewer. In practise, the value is always a small amount after the 64bit FileTime based last accessed time.

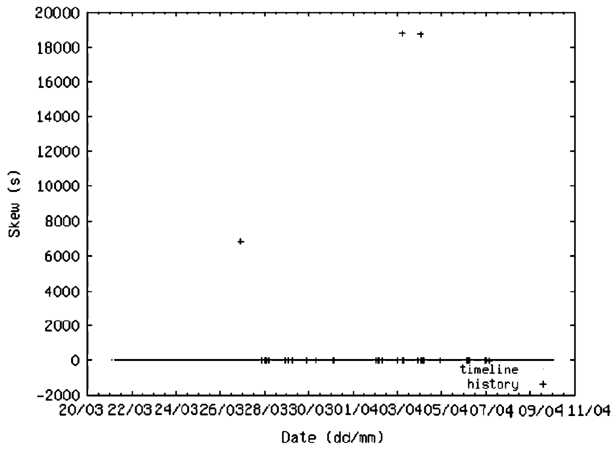

Fig. 11 – Incomplete information.

log file, resulting in a far greater offset. Adding the excluded log produces the results seen previously in Fig. 10.

There is another problem related to the closed world assumption. By assuming that all data are present, we assume a perfectly functioning logging system in Internet Explorer and Squid.

The corollary of this assumption is that we assume a perfect implementation for our *index.dat* parser and that we have interpreted the semantics of the records correctly, and also avoided bugs in our implementation. Clearly, this is not valid given the challenges in reverse engineering the file format and allowing for inevitable bugs. We expect that the false positives present in the history correlation algorithm are attributable to these.

Despite the challenges outlined above in correlating IE History and Cache records with squid access logs, we find that we are able to correlate the records to a dataset which correlates reasonably with the time streams sampled from the first experiment.

We expect far higher rates of true positives are possible; both algorithms ignore large parts of the dataset, as our heuristics and rules only apply to a small proportion of the dataset where we can infer certainty in matches. Algorithms which model uncertainty in matching records and incorporate probabilistic methods hold promise towards this goal; we are currently investigating Monte Carlo Markov Chains as a potential approach. We expect that the principles underlying Gladyshev and Patel's (2005) event bounding approach could have relevance.

We have observed decreases in the rate of false positives with the elimination of a software fault or correction of a misconception in semantics. We expect that the rate of false positives will decrease with further investigation and understanding of the *index.dat* file format.

5. Conclusions

In this paper, we have investigated two themes related to uncertainty in interpreting timestamps. Firstly, we investigated whether it is reasonable to assume uniform behaviour of computer clocks over time, and attempted to characterise how computer clocks behave in the wild. Secondly, we investigate the feasibility of automatically identifying the temporal behaviour of a computer by correlating timestamps embedded in digital evidence with corroborative time sources.

By logging the relationship between system time and corresponding civil time, we have identified a number of instances where the timescale of three computers behaved in anomalous and indeterminate ways. We conclude that the interpretation of timestamps from digital evidence is complicated by the potential for unpredictable temporal variations.

The implications of this for forensics are clear. In cases where establishing the precise time at which a computer event occurred is important, one cannot assume that computers running MS Windows 2000 or XP have behaved in uniform ways with respect to keeping time.

In this study, we have focused on the Windows platform due to its dominance in deployment. The extent to which the results relate to temporal behaviour of other operating systems is still an open question. The computers in this experiment were tethered to a time source where synchronisation occurred often. Future work is needed in characterising the behaviour of Windows PCs that are either untethered from or loosely tethered to reliable time sources.

Finally, we have investigated the feasibility of using a ubiquitous source of timestamps – those contained in browser records – as a partial means of establishing the temporal behaviour of a computer over time. Two algorithms were proposed and evaluated, and experimental results were presented which demonstrate that both algorithms produce outputs which correlate reasonably with the timescales of the subject computers. We have additionally described how the history correlation algorithm could be modified to produce a higher rate of true positives.

There are a number of areas where future work is warranted. First, in order that results based on this type of correlation may be more clearly interpreted and explained in forums such as courts of law, a means of qualifying and quantifying the error involved would be of use. Second, in order that the resolution of the characterised timescales may increase, improved algorithms which incorporate uncertainty in record matching should be investigated. Finally, the Internet Explorer *index.dat* file format is still not fully understood. We expect that a clearer understanding of the file format would lead to a reduction in errors.

Acknowledgements

We thank our colleagues Jason Smith, Mark Branagan (ISI) and Julien Vayssiere (SAP Research) for their contributions and suggestions in reviewing this paper.

References

Boyd C, Forster P. Time and date issues in forensic computing – a case study. Digital Investigation 2004;1:18–23.

Gladyshev P, Patel A. Formalising event time bounding in digital investigations. International Journal of Digital Evidence 2005; 4(2).

Jones KJ. Pasco – an Internet Explorer activity forensics analysis tool, http://sourceforge.net/project/shownotes.php?group_id=78332&release_id=237810; 2004 (accessed April 2006).

Microsoft Corporation. How Windows keeps track of the date and time, http://support.microsoft.com/?kbid=232488; 2006a (accessed April 2006).

Microsoft Corporation. The system clock may run fast when you use the ACPI power management timer as a high-resolution counter on Windows 2000-based, Windows XP-based, and Windows Server 2003-based computers, http://support.microsoft.com/?kbid=821893; 2006b (accessed April 2006).

Microsoft Corporation. Microsoft products do not reflect Australian daylight saving time changes for the year 2006, http://support.microsoft.com/kb/909915; 2006c (accessed April 2006).

Mills DL. Precision synchronization of computer network clocks. ACM Computer Communication Review 1994;24(2):28–43 (April 1994).

Mills DL. A brief history of NTP time: confessions of an Internet timekeeper. ACM Computer Communications Review 2003; 33(2):9–22 (April 2003).

Nolan R, O'Sullivan C, Branson J, Waits C. First responders guide to computer forensics. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University; 2005.

Sandhill Consulting. How Microsoft Windows NT 4.0 handles time, http://folkworm.ceri.memphis.edu/ew-doc/PROGRAMMER/NTandTime.html; 1998 (accessed April 2006).

Stevens MW. Unification of relative time frames for digital forensics. Digital Investigation 2004;1:225–39.

Thomas LK. Reverse engineering index.dat, http://www.latenighthacking.com/projects/2003/reIndexDat/; 2003 (accessed April 2006).

Weil C. Dynamic time & date stamp analysis. International Journal of Digital Evidence 2002;1(2).

Bradley Schatz is a doctoral candidate at the Information Security Institute at the Queensland University of Technology, Brisbane, Australia. His research focus is digital forensics, knowledge representation, and event correlation. Prior to his entry to research, Bradley practiced software engineering, IT security and systems management in industries such as banking, entertainment and health.

George Mohay is an Adjunct Professor in the Information Security Institute at the Queensland University of Technology, Brisbane, Australia. Prior to that he had been Head of the School of CS and SE from 1992 to 2002. His current research interests lie in the areas of computer security, intrusion detection, and computer forensics. He is on the Program Committee for RAID, *Recent Advances in Intrusion Detection*, and is on the program/steering committees of a number of other international conferences. He is General Chair for RAID 2007 to be held on the Gold Coast, Australia.

Andrew Clark is a Senior Research Fellow in the Information Security Institute at Queensland University of Technology. He obtained his PhD in 1998 in the area of cryptography and is currently researching actively in the fields of intrusion detection and computer forensics. He supervises numerous postgraduate research students in these areas and is also an active participant in industrial projects with government and corporate partners.