

Event Sequence Mining to Develop Profiles for Computer Forensic Investigation Purposes

Tamas Abraham

Information Networks Division
Defence Science and Technology Organisation
PO Box 1500, Edinburgh SA 5111, Australia
Email: tamas.abraham@dsto.defence.gov.au

Abstract

Developing profiles to describe user or system behaviour is a useful technique employed in Computer Forensic investigations. Information found in data obtained by investigators can often be used to establish a view of regular usage patterns which can then be examined for unusual occurrences. This paper describes one such method based on details provided by events found within computer forensic evidence. Events compiled from potentially numerous sources are grouped according to some criteria and frequently occurring event sequences are established. The methodology and techniques to extract and contrast these sequences are then described and discussed along with similar prior work in the same domain.

Keywords: Data Mining, Computer Forensics, event sequences

1 Introduction

Computer Forensics is an emerging discipline investigating computer crime. It assists investigators in the analysis of often large amounts of data by, for example, automating trivial processing activities. Additionally, computer forensic software is often designed to find clues to follow-up leads found in data. These *suggestions* can be generated automatically using information found in the data, often guided by background information supplied by investigators, such as their knowledge and expectations about the contents of the data. Evidence related to a shortlist of suspicious activities condensed this way can then be further investigated in more detail (Casey 2000). Importantly, Computer Forensic software is not designed to solve crime, but to assist in their investigations. The primary goal of these systems is to reduce investigation time and complexity.

Investigative profiling is an important activity in computer forensics that can significantly narrow the search for the perpetrator and reason about the perpetrator's behaviour. This is analogous to criminal profiling which focuses on establishing personality characteristics of an offender in order to identify the type of person involved in the crime under investigation (e.g., arson). Profiling can also aid in identifying the type of activity the perpetrator is engaged in e.g., e-mail authorship analysis may identify the educational level or gender of the offender and may, consequently, be able to establish if an e-mail

has been masqueraded (de Vel, Anderson, Corney & Mohay 2001).

Computer Forensics utilises techniques shared by a variety of other computer security related disciplines: however, computer forensics is chiefly performed on static data sources (it is a type of 'post-mortem' analysis). Data Mining has been successfully used in a variety of domains with large data sets. It has proven useful for a number of purposes, for example, describing the contents of data with patterns to summarise the characteristics of the data set even in the presence of noise and incomplete data (Ester, Kriegel, Sander & Xu 1996). Patterns can also be viewed as descriptions of the behaviour of the data and can be the basis of the development of profiles (a set of statements about the behaviour of what is being investigated), an important resource for analysis in Computer Forensic investigations.

Data mining for the more specific purpose of constructing personal profiles has been used in the context of customer personalisation. Here, marketing content and services are tailored to an individual on the basis of knowledge about their preferences and behaviour. Applications include content-based and collaborative filtering-based recommendation systems, customer profiling (Adomavicius & Tuzhilin 2001, Adomavicius & Tuzhilin 2001b, Hirsh, Basu & Davidson 2000), fraud detection (Fawcett & Provost 1997), and web browsing activities (Chan 1999, Nanopoulos, Katsaros & Manolopoulos 2001, Tan & Kumar 2001, Mobasher, Dai, Luo, Sun & Wiltshire 2000). Content-based recommendation systems model the link between data content and a person's preferences for that content whereas collaborative recommendation systems model the link between a person's preferences and other persons' preferences for the given data content (Konstan, Miller, Maltz, Herlocker, Gordon & Riedl 1997, Nesbitt & de Vel 1998). Another relevant activity within Data Mining is the extraction of sequential patterns (Agrawal & Srikant 1995, Mannila, Toivonen & Verkamo 1997), which analyses temporally ordered data in order to model repetitive behaviour.

In this paper, we describe a technique to profile and analyse computer forensic data based on events. We use an event chain analysis approach not yet employed in this application domain, adapted where necessary to accommodate the particular environment. Earlier attempts by the same author(s) using the association mining paradigm and generalisation are documented in (Abraham & de Vel 2002, Abraham, Kling & de Vel 2002). This paper shares the same motivation and similar methodology but handles data from multiple sources and uses a different set of tools to achieve its goals, while also adding some new ideas to the previous presentations. In Section 2, we introduce the concepts used in our approach to computer forensic profiling. Section 3 contains the definitions

we use and how they relate to our approach. Tests performed on computer forensic data are detailed in Section 4, followed by our conclusions in Section 5.

2 Concepts of Investigative Profiling using Events

The goal of developing profiles on the data available to Computer Forensic investigators is to provide insight into the day-to-day operation of the computer environment being investigated. This process, whilst utilising established data mining techniques and methodologies, can be influenced by a number of factors particular to computer forensics. The most relevant factor in this paper is the concept of a *subject*, represented by distinct values occurring in the data which are used to sub-divide an overall profile: a profile being established on the complete set of data gives general information about the data, but it also consists of sub-profiles that are particular to certain elements of the data. For example, a profile that contains summary rules about users logging onto a server from various clients will be able to be broken up into sub-profiles where rules involving a particular user (as represented by distinct values of a ‘user’ attribute) or user group can be treated as their own specific login profiles. Within the same rule set, other ways for breaking up the general profile can also be based on, for example, day of the week, or location of the client. Computer forensic investigations are instigated due to demand, which allows for targeting specific needs and thus constraints can be put on the technical implementation to cater for multiple processing approaches.

Introducing subjects into an investigation also carries the added task of reconciling data from various sources onto the right subjects. User names from logs on different computers, for example, may belong to the same person, and these same people may also be identified differently in additional sources of information, such as security door logs, e-mail addresses and so on. Establishing such mappings, however, are often a natural part of the investigative process and can also be automated to some degree (for example, Unix like operating systems often contain the full names of users in their password files which can then be compared to other data). There is, however, little doubt that pre-processing data for profile development can be a complex and time-consuming part of an investigation.

Often, a simple data mining technique can compile a profile (here meaning a collection of subject-related sub-profiles) effectively. In our earlier paper (Abraham & de Vel 2002), association rule mining (Agrawal, Imielinski & Swami 1993) was used to generate rules on login data, although in that paper it was not specifically underlined that the resulting rule set was in fact a subject profile collection. The paper, on the other hand, extensively discussed another aspect of profile analysis, namely the contrasting of profiles. There are two parts to this kind of analysis, that of comparing profiles *to each other*, and that of looking into the contents of each profile to find discrepancies *within* the profile. In the first instance, differences between separate subject profiles are usually expected and are indicative of the individual behaviour of each profile subject. It may be possible to assign a measure (however subjective that may be) to the difference between individual profiles, either as a profile total or as an indication for respective differences between individual profile elements. On the other hand, when investigating several subjects, observing the similarity of their profiles is far more useful. It may, for example, be possible to group

individual profiles together based on their similarity and investigate subjects whose profiles may lie outside major groups. These groups may also help delineate alternate behaviour as often general profiles (the collection of subject profiles) will contain a number of strong rules/observations that may contradict each other, when in fact each may be a defining, distinct characteristic of a given group. The same kind of alternate behaviour within a subject profile, on the other hand, may have a different meaning and rather than offering an explanation, may indicate something unusual. Apart from needing to investigate infrequent behaviour in a subject profile (known as outliers), alternate behaviour for a given subject may indicate the presence of two separate subjects acting under the same identity. The selection and application of appropriate measures to distinguish between normal and unusual behaviour therefore becomes highly important.

2.1 Representation of subject behaviour

Forensic evidence comes from a multitude of sources and varies in available content, such as the granularity and completeness of information about the actions being recorded. A simple door log may contain not much more information than a time of entry, a card identifier and door descriptor, whilst data extracted from an e-mail header, for example, can not only identify sender and recipient, but the path the e-mail has taken during its route, the mail transfer agents used, the subject of the letter and so on. To effectively organise and mine data of such varying nature, a common representational framework can be employed. The simplest of these may include information describing the *subject* (the attribute whose distinct values cause the separation into sub-profiles), the *action* performed, the *object* involved in the action (if any), and a *timestamp*. This 4-tuple corresponds exactly, for example, to the above mentioned door log representation. In the case of e-mails *sent*, the sender would be mapped to subject, recipient to object, the action itself would be the process of an e-mail being sent, with the timestamp corresponding to, for example, time of dispatch. When e-mails are *received*, the mappings would be reversed: the subject would be the recipient, the object the sender, the action the process of receiving an e-mail, and the timestamp the time of arrival.

It is quite feasible that most evidence collected during a computer forensic investigation and found in log files and other sources can be transformed into a single representation described above. The core attributes can be said to *define* an event. Additional information regarding the actions being stored can either be placed into auxiliary tables, or pointers can be added to the main table to refer back to the source where they can be found. These bits of information can be useful in later analysis when drilling down into particular details. The main technical challenges during the compilation of event data (apart from the already discussed mapping of identities) include

- the ability to parse multiple format sources, and
- reconciliation to a uniform timeline.

The first of these challenges can be addressed by programming for each available source. This requires the creation of a group of parsers designed to cater for a single or a family of similar sources. Generic parsers capable of handling a number of sources are widely available and can be used for this purpose (Abbott 2004). Extracting the subject, object and timestamp information can be as trivial as passing the

correct attribute designators to these parsers. Identifying the correct action, however, can be more complicated, and requires an additional control mechanism. Not only is there a need to have parsers that can handle the various types of evidence, but they also need to be able to distinguish the type of action they are recording each time. This means that the parser should only record actions that can later be uniquely recognised. For example, a door log may represent entry through a particular door using the value “in”, and exit through the same door using the value “out”. Another door log may do this differently, for example, by using the words “Entry” and “Exit”. Clearly, they represent the same actions and should therefore be stored with the same action *identifier* or type. This necessitates the need to have a uniform notation for actions: for example, a list of accepted types. Parsing will be more complicated (it may involve translating values read from file into accepted types), but will ensure consistent and unique representation in the resulting table (Abbott 2004).

The second problem pertinent to handling multiple-source timestamped data is the difficulty in establishing a uniform timeline. To be able to reconstruct a particular sequence of events as they happened, those events need to be ordered correctly as a sequence in the reconciled action/event table. This might be a complex operation where each source needs to be associated with its own time representation, or ‘clock’. Common representations used include Universal Time, Coordinated (UTC) and local time zones with or without employing daylight saving time. Often, hardware clocks are incorrectly set on some computers and these discrepancies need to be identified and remedied. Thus, an integral part of data preparation prior to extracting events from evidence is the development of a model to accurately represent time across the heterogenous source environment. In some cases, this process may be straightforward and fully automated, in others, investigators will need to manually identify, enter and adjust model parameters (Stevens 2004).

Once event evidence has been stored in a table associated with subjects, event sequences can be constructed. The simplest method to express these sequences is in the form of chains (such as Event $A \rightarrow$ Event $B \rightarrow$ Event C , or ABC for short) with elements representing single events. These chains are constructed for each subject (or subject group) using the timestamps in the data to establish the correct order. The events themselves may comprise more than just the action taken at the given time: they may incorporate object information, the outcome of the action and so on. For example, an entry in the database for X (subject) describing a login (action) on computer Z (object) that failed due to an incorrect password (result) may actually be the event “failed login on computer Z” for the given subject. It is up to the algorithmic solution to construct such events using available data in the action table.

2.2 Extracting sequences from reconciled event data

Mining frequent episodes in data is a well-researched area of data mining, with a large body of work in existence. For a recent discussion of some of the earlier work and a current algorithm, refer to (Pei, Han, Mortazavi-Asl, Wang, Pinto, Chen, Dayal & Hsu 2004). To accommodate the mining of such episodes for Computer Forensic profile building purposes, the following characteristics are desirable:

1. Relative ease of setting up profiling. The major parts of this process are the construction of

a source table for mining, discussed above, the setting up and running of a set of parsing algorithms and the building of a time model.

2. Acceptable resource and time requirements for the profiling algorithm. Considerable time may have already been spent setting up and creating the source table, and therefore results should be available in a reasonably short amount of time thereafter. Efficiency is quite important, and a solution that only needs to read the database table once (reducing I/O requirements) is preferable. Computing resources are usually adequate to handle the large memory requirements typical of such one-pass algorithms as processing power and storage space is essential in a computer forensic investigative environment and this often implies that large amounts of main memory are installed. Often, an upper bound to sequence length can be specified to control memory requirements without affecting analysis outcome greatly (long event sequences often do not represent habitual behaviour and are therefore not essential in profiles). Analysing a profile stored in memory may also be advantageous efficiency-wise.
3. Completeness of profile. A major disadvantage of a large number of data mining algorithms from a computer forensics perspective is that these algorithms generate descriptions according to some metric. Data not supported above a certain threshold is often not expressed in the resulting rule set and therefore finding these outliers can only be achieved by revisiting the original data set and comparing it to the rule set. If outliers were included in the profile, this could be avoided, albeit at the added cost of having a much larger sized profile.
4. Period-specific sequence generation. Often, behaviour follows a pattern that repeats itself periodically, typically on a daily basis. This needs to be handled by the profiling solution upon request.
5. The facility to extract regular patterns. As behaviour is often repetitive, some patterns may repeat themselves often, but with some variation. For example, a subject may enter a restricted work area (door log event), log on to a computer (login event), read e-mail (pop event) and browse an online newspaper (HTTP event) every day. These events also tend to occur within a short period of time, but occasionally may not follow the exact same sequence. For example, the subject may decide to grab a coffee from the outside kitchen area after logging in, and re-enters his restricted work area again before continuing to read e-mail and the newspaper. Such an event will disrupt the daily routine but in essence will not change it, and this needs to be taken into consideration by the profiling algorithm. For example, there may be a strong (highly supported) sequence ABC and another (possibly weak) sequence $ABDC$, where the addition of event D should not necessarily detract from the strong, more general gapped sequence $AB * C^1$.
6. Adjustable sequence length. Sequence length may vary according to requirements but needs

¹A more general approach to this problem would be based on the use of sets. We could argue that the order the events occur can also vary. That is, the sequence ABC may not be different to ACB or even BCA . However, this study is outside the scope of this paper as it relaxes one of the main criteria investigated, namely the ordering of a sequence itself.

to be able to be controlled, for example, by using a temporal sliding window. A maximum sequence length should also be available in case this window is not sufficient to control the length of sequences being generated.

2.3 Event chain profile development summary

The main tasks in preparing and building computer forensic event data into profiles can be described as follows:

- Events are created from raw data by first using parsing algorithms to format raw input into a set of attributes, and then they are defined as a composite result of one or more of these attributes. Events may be attached to a subject (usually just another attribute extracted by the parser), and can have a number of additional event-specific attributes that further describe the event but are irrelevant in *defining* the event (e.g. if both the username and userid are in a record, one is used to uniquely identify the subject, the other is irrelevant in defining the event but can be used to portray additional information about the event). The event building process takes into consideration the following criteria:
 - The reconciliation of events to particular subjects (using mappings of alternate subject representation, if necessary); not always needed but useful (e.g. if we deal with with user names) as it allows the creation of separate event chain collections.
 - The use of a unified timeline to create an accurate temporal ordering of multiple-source event data.
- The event profile generation algorithm creates a set of statements about the events extracted from raw data. Its characteristics include:
 - Generates event chains potentially associated with individual subjects.
 - The length of chains is controlled by the user or a strength-based variable length strategy is employed.
 - The use of background knowledge, often used in data mining algorithms, is limited to mappings as mentioned above.
- The final task of profile generation and analysis is the comparison of chain sets in order to establish similarity between subjects. If there are similar relevant patterns repeating for some subjects, they are exhibiting similar traits and are therefore more closely related to each other than others without similar patterns.² For both subject-related event chain collections, and the global chain set (i.e. the event chain pool without subjects), intra-chain differences can also be found to detect alternate and irregular behaviour patterns.

3 Definitions and implementation

The previous section introduced the concepts of the event profiling process. In order to create and analyse profiles, raw source information needs to be organised and transformed into formatted data. Each source data set can be parsed into a relation R_d containing

attributes $A_i, i = 1, \dots, K_d, d = 1, \dots, D$, where D represents the number of distinct source data types, K_d the number of attributes for a given source d . In real-life implementations, a relation can correspond to a database table. Let the number of records in table d be denoted by N_d . An *event* E is the n -tuple $\cup_i a_i$ for some attributes in relation $R_d, i = 1 \dots n_d$, where n_d may vary for different types of relations, $n_d \leq K_d \forall d$. Each attribute a_i corresponds to one attribute A_j of R_d with no repeats, $j \in [1, \dots, K_d]$. A *unique event* is an actual distinct value of an event E found in the data. An *event sequence* is the ordered listing of unique events of some length. The sequence $E_1 E_2 \dots E_k$ containing k events is a k -sequence or k -chain. The events $E_i, i = 1, \dots, k$ may contain repeats of the same unique event(s). Naturally, a 1-chain describes a single event.

A *subject* s_j is a value of a special attribute $a_s, s \in 1, \dots, K_d$, where a_s may or may not be in the n -tuple $\cup_i a_i, j = 1 \dots M_d, M_d \leq N_d$ (usually $M_d \ll N_d$, that is, the number of distinct values in the subject attribute is much fewer than the number of records in the relation). If subjects form part of the events, mining and analysis is performed globally, otherwise events are attached to subjects. In this case, the original relation R_d may be divided into subsets $R_d^j, \cup R_d^j = R_d$, so that R_d^j contains only events where the subject is s_j . An event may have additional (non-event defining) attributes attached that provide additional information about the event. One of these must be a *timestamp* t used to establish event order.

Events are constructed according to the above definitions, using available relations and rules on how to combine attributes into events. For example, an event could be ‘Successful entry through door X ’, another could be ‘Unsuccessful exit through door Y ’. Here, attributes *Success* (with values “yes” or “no”), *Entry direction* (“in” or “out”) and *Door* (an identifier) may be used to produce a triplet into a unique event. The potential number of combinations depends on the number of actual attribute values appearing in the relation. The ‘*event rule*’ here is represented by the triplet $(a_{Success}, a_{Entry}, a_{Door})$. Actual triplet values, or unique events, can be stored in a table and added to as they are discovered on-the-fly through composition from records in relation R_D . A unique identifier is assigned to each entry in the table (e.g. event A, B , and so on).

3.1 Importance measure

A combination of i unique events makes up an event sequence S of length i for an arbitrary $i > 0$. Let T^i be the total number of event sequence occurrences (‘routes’) of length i in the data set, and R^i the total number of unique routes of length i observed in the data. Let r_j^i be the route count for route $j \in [1, \dots, R^i]$ at level (length) i . The significance s_j^i of route r_j^i is then defined as $s_j^i = r_j^i / T^i$. Note that $\sum_{j=1}^{R^i} r_j^i = T^i$, hence $0 \leq s_j^i \leq 1$.

A minimum significance threshold (usually something quite low, e.g. 0.05) needs to be reached for a route to be classified as ‘not unusual’. Significance can also be ordered and the assignment of the level of how unusual a route is can be automated (for example, we could say that routes in the lowest X percentile are unusual). On the other hand, ‘interesting’ routes should have a significance that is relatively high (e.g. > 0.25).

²As a side note, this is also a way of abstraction to higher level without background knowledge.

3.2 Unusual events

Unusual events are rare events, that occur much less frequently than others. This is equivalent to stating that the event has a low significance (or alternatively, a ‘route’ of length 1 has low significance as specified by some user defined threshold). Similarly, we can have unusual event sequences.

3.3 Sequence generation stopping criteria

Sequence generation can be stopped or re-initialised according to pre-defined criteria. They include

- Sequence length. The sequence cannot be longer than a given positive integer.
- Time. Sequence generation can be re-started due to time passing (day, week etc).
- Particular occurrence. An event (or event sequence) occurring may trigger the re-start of event sequence generation, for example, a subject exiting the main building.
- Other user and situation specific criteria.

3.4 Sequence generation algorithm

Having determined the requirements on the representation and process of generating event sequences for a given subject, the algorithm to produce a sequence tree then becomes a two-fold process. The first part of the algorithm formats data into events, the second uses a single pass over this data to build the tree. An example event sequence tree is found in Figure 1.

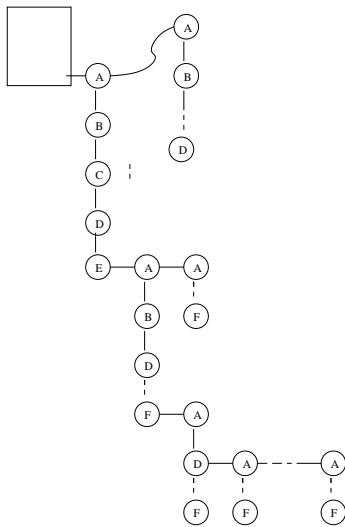


Figure 1: A subject event sequence tree

Note how the tree in Figure 1 contains nodes of single events, with each node having its own sub-tree (subtrees are found to the right of a node). At each level, all events that occur after the previous events (that is, in the earlier part of the sequence) are collected in a list. These nodes are represented vertically in the figure. This means that at the top level of the tree, each event involving the subject will be listed. This representation may be able to be compressed; see discussion in Section 4.2.

The algorithm to produce a subject event sequence tree contains two parts. The first part prepares the data, and is heavily dependent on the source information and selection of subject and timestamp fields. The second one makes a single pass over this newly

compiled event data and builds the tree. The steps performed are as follows:

Algorithm 1: Subject Tree Builder

Inputs: Source data containing event details; subject and timestamp field specifications; sequence generation stopping criteria; maximum sequence length k

Outputs: A collection of subject event sequence trees

1. Pre-process:
 - 1.1. Detaching subject and timestamp fields, convert data into unique events
 - 1.2. Group events by subject, order each group by timestamp
2. For each new subject,
 - 2.1. Create blank tree
 - 2.2. Initialise current length $l := 1$
 - 2.3. While more events for subject
 - 2.3.1. Fetch next event
 - 2.3.2. Increment l
 - 2.3.3. If sequence generation stopping criteria, re-initialise length l
 - 2.3.4. Add event to tree up to l times
3. Stop.

3.5 Minimal non-repeating sequence length

By using the event notation described before, the sequence $ABAB$ is an example for a 4-length sequence. However, it contains a repeat of the unique 2-length sequence AB . This raises the question on how to handle repeating sequences: what sort of strategy should be employed in treating sequences that include repeating subsequences but otherwise do not trigger a sequence generation stopping criteria? How much validity can a repeat have in forming part of an important longer sequence? Common sense dictates that a long sequence that contains regular repeats of a shorter sequence should be instead treated as multiple occurrences of the shorter subsequence. Thus, it stands to reason that only sequences that contain no repeats truly express regular repetitive behaviour. On the other hand, what if these repeats are encapsulated between important events that do not otherwise occur together in a sequence (for example, $ABCBCD$, where A and D might not otherwise be part of a sequence together). The strategies to handle such situations may include the following:

- Defining a maximum repeat length. We may decide on an ‘atomic’ size for sequence. By default, an event, the smallest part of the sequence, is atomic. However, for repeats, this may be too small. For example, is the above $ABAB$ repeating? It probably is, as it contains the same short sequence AB twice, but then, is ABA repeating? If we use a single event as the atomic size, then it would be. In some cases, this is not very useful. For example, if A signifies going through a door (in either direction), then ABA could be a ‘complete’ event sequence. Maximum repeat length then establishes how long a sub-sequence can be before its re-occurrence in a sequence is considered a repeat. A maximum repeat length of 0 means no single atomic event can be repeated in a sequence (i.e. ABA is repeating), length 1 means single events can be repeated (i.e. ABA is valid, even AA , but not AAA).
- Preferring event chain uniqueness. An invariant observation is that if all event sequences are generated up to a given length, then even if one chain is removed due to repeats, all information about the smaller unique chains that it included are still available separately. That is, no relevant information is lost, and hence the longer, repeating chain may still be considered as a useful source

of information. This strategy suits the goal of being able to extract regular patterns from similar sequences, as discussed earlier.

3.6 Longest non-repeating sequence

Regardless which strategy is preferred on repeating sequences, an interesting related problem is that of finding the longest regular non-repeating sequence for a given subject. This sequence, when supported sufficiently, can be a strong characteristic for the specific subject as it describes a routine-like event sequence with varied content. Such sequences can be considered ‘traits’, or defining characteristics. Therefore, it is important to be able to find such sequences during profile generation. Since it is not known *a priori* what length these sequences may have, it could be a good strategy to profile with a sufficiently large maximum sequence length. This may, however, result in longer sequences having repeats, so that it may be necessary to post-process them to remove the ones with repeats.

An algorithm to establish if an event sequence S of length $n > 1$ contains repeats can be as follows:

1. Create all sub-chains R_i^j of S for lengths $j = 1, \dots, \lfloor n/2 \rfloor$, $i = 1, \dots, m^j$, where m^j is the number of distinct subsequences at length j .
2. For $j = 1$, remove S as a repeating sequence if any R_i^1 (i.e. a single event that is part of the sequence) occurs twice in a row, that is, at least two occurrences are neighbours in the sequence.
3. For $j > 1$, remove S as a repeating sequence if any R_i^j occurs at least twice.

Having established non-repeating sequences for each subject under profiling, it gives us the opportunity to investigate these subjects for similarity based on these sequences.

3.7 Subject similarity

Similarity of subjects is defined by similarity in behaviour expressed in their profile. In this case, this is the collection of event sequences belonging to each subject. The advantage of finding similar subjects in a collection is that this enables the creation of groups within the total profile: subjects that are similar can be clustered together and thus distinct sub-groups within the profile can be established. Grouping subjects adds another level of abstraction and allows additional analysis to take place, for example, in the form of group-level outlier identification.

Subjects can be considered similar based on some pre-defined similarity threshold. One common way to establish groups is to start with individual subjects and pair them to form groups and merge these further. Alternatively, a single large group can be separated into smaller ones. The approach preferred usually depends on the measure used, which, in the case of event sequence profile data, must be established so that only significant profile data is compared. That is, unusual events for individual subjects should be ignored, because even though they contribute to the full description of a given subject, they cannot be considered as one of their traits, or defining characteristics. Some repeating sequences should also be ignored, as they can be considered more as examples of habitual behaviour consisting of smaller length defining traits (see previous section). This typically leaves a set of short (of lengths 2-4) important non-repeating sequences, which form the basis of the following strategies to find sub-groupings:

- Grouping is based on similar event sequences using a similarity threshold or grouping model.
- Individuals within groups share the same traits but can have other traits that are not group ones. These non-group traits of individuals within a group can be important for further investigation.
- Individuals also have outlier behaviour. These could be sequences that may or may not be traits in another group.
- Non-trait behaviour (i.e. sequences that are not significant) can be contrasted to trait ones with an appropriate measure to quantify distance, or irregularity from the observed normal behaviour exhibited by *all* groups.

3.8 Pruning

The algorithm used to generate subject-specific event sequences produces a complete, lossless representation of all sequences observed for the subject up to a pre-defined length. Within each set, a number of sequences will be significant according to some measure (traits), others will be less relevant but still important (interesting), with the rest having negligible support (outliers). Some of these sequences will be very similar, either due to repeats within longer sequences, or the insertion of irregular events into an otherwise significant, regular event. Pruning, in this case meaning the process of reconciling these variations to produce a compact set of sequences by abstracting similar sequences higher to a more generic sequence description, can be employed to compact the profiles further. Often, outlier sequences are variations of more significant event chains. For example, the behaviour exhibited by the two sequences ABC and its variant $ABAC$ may be replaced by the combined expression $AB * C$, where ABC was significant, and $ABAC$ an outlier. This represents adding another, higher level of expression to sequence representation. The approach for pruning from this perspective requires the following:

- Combining to more generalised expressions is a depth first approach. That is, the longest available sequences are investigated first.
- To combine two sequences, one would have to be longer than the other³. In practice, this length difference will be 1 due to the condition above.
- The longer event chain is preferred to be an outlier or a less significant sequence (the shorter one can be anything as it may get processed later). There is no point in combining traits even if there are quite similar, as each occur often enough to warrant separation.

3.9 Event types, ranking and subject groupings

The next aspect of event sequence handling to consider is event granularity. To be able to group subjects successfully, a less rigid definition for events may have to be used. That is, if two events are very similar (e.g. ‘logging on to server A’ may be similar to ‘logging on to server B’), then a hierarchical grouping of events may be employed to find a set of ‘higher level’ core events upon which chains can be developed. This introduces the concept of event *types*, where events

³It may be possible to combine same-length sequences. This would, however, require an abstraction between unique events, that is, the development of event groups (such as events A and B being replaced by a super-event A'). See Section 3.9 for further details.

can be organised into several layers of abstraction and event sequences may be constructed using these composite event designators. This approach is similar to the use of concept hierarchies often employed in other data mining approaches (Abraham et al. 2002).

Grouping of subjects can occur based on event chains at a given length. For example, ranked 1-chains (events) can be used to find all users that perform the same event. *Ranking* of event chains can be calculated at each length and this ranking can then be applied to assign individual subjects to groups based on their most significant trait, or event sequence at that length. Ranking events chains of different lengths together is not practical; shorter chains by definition are at least as significant than longer ones based on them. Using rankings compiled at different lengths, however, is possible. For example, groups can be created at level 1. Then each of these groups can be broken up into subgroups based on rankings at level 2 *within* each group. This procedure can be repeated up to a preferred maximum sequence length, breaking each subgroup up further. In some cases, some members of a group may not be able to be assigned into a subgroup and hence remain elements of the ‘super’-group. After all sub-groupings are completed, we are left with a hierarchical group representation, where each leaf node will contain some subjects, as well as some higher level nodes that were not completely distributed into subgroups. The contents of these non-empty nodes gives us a distinct distribution of subjects based on their behaviour as observed in the event data.

The algorithm for this approach can be implemented as follows:

Algorithm 2: Subject Grouping based on Event Rankings

Inputs: Subject event sequence trees; Ranked event sequences; maximum sequence length k

Outputs: A collection of subject groupings

1. Initialise ‘level 0’ group to all subjects
 2. For each chain length up to the maximum
 3. For each existing grouping on this level
 4. For each subject in this group
 5. If subject has significant trait on this level
 6. Move it to most significant trait subgroup
 7. (create subgroup if it does not yet exist)
-

Variations on the above algorithm may exist. For example, at each level, there may only be a small number of subgroups allowed. That is, only significant longer event sequences will be used to generate further partitioning of the group; a subject with no particular significant trait at this deeper level will remain in the super-group.

3.10 Events without subject

In some cases, events may not be attached to subjects, but treated globally. This approach is useful when sequences are created to profile the environment based purely on events, where each subject is just another attribute of an event. This allows investigating a profile from a different perspective: to find recurring attribute patterns within frequent event sequences. Usually, the attribute we would be interested in is in fact the subject itself. For example, if investigating the short event sequence AB in the profile shows that when the attached subject of event A is X , then 75% of the time the attached subject of event B is also X , (or any other subject, Y , for example), then we have found a strong pattern within the sequence. This illustrates how non-subject attached event sequences can be further (meta-)mined

for inter-subject relationships, although this is not part of the main discussions in this paper.

4 Testing

To demonstrate the algorithms and methodology described in this paper, we performed some testing using a real-life data set. The test illustrates the profiling algorithms by producing a grouping of subjects based on their regular behaviour patterns as observed in the data.

4.1 Door Log Test

The test uses data obtained from door swipe access logs in an office building that houses just over 100 people. Data were obtained from three doors over a period of one calendar month: at the main entry door where everyone entering the building has to swipe both in and out, and at two section doors, one of which only requires swiping on the way in, and the other both ways. People who have access to the section door that requires bidirectional swiping also have access through the other section door, but not vice versa. Figure 2 illustrates the access rights groups of workers have for entering sections of the building.

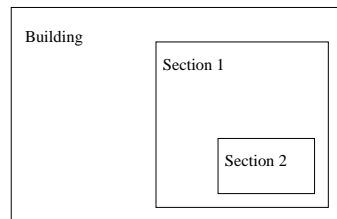


Figure 2: Access rights topology for office building

The typical events and event sequences that could be observed in the data (having removed subject identifiers and timestamps) can be seen in Tables 1⁴ and 2.

Event	ID	Frequency
Entry through Building door	A	20.68%
Exit through Building door	B	20.40%
Entry through Section 2 door	C	20.06%
Exit through Section 2 door	D	20.10%
Entry through Section 1 door	E	18.76%

Table 1: Individual door access events

The data set used in this experiment limits the range of conclusions that can be drawn. Short event sequences that occur frequently can be trivially deduced assuming subjects swipe as intended. The number of common event chains of lengths longer than 2 or 3 events are limited, as they start becoming random and lose significance (for example, people with access to Section 1 may move within the building and repeatedly swipe event E).

The main reason to perform this test is to validate the grouping strategy developed for subjects based on significant event sequences found in their profile. A complete list of subjects with access rights to different parts of the building was obtained and compared against sample runs of Algorithm 2. Figure 2 shows that in reality there are 3 major groups according

⁴Note that even though some events (entry and exit pairs) are symmetrical, the percentages of corresponding events may not match. This is due to the common practice of ‘shoulder surfing’, that is, not swiping when the door is opened by someone else.

Event ID Sequences
AB
AC
AE
DB
AEB
ACD

Table 2: Typical frequent short door access event chains corresponding to movement paths

to door access rights. Algorithm 2 produced 5 groups that correctly placed every subject in the three major groups: there were, however, two additional groups with small memberships that contained exceptions. These were people with the following characteristics:

- They were occasional visitors to the building or regular workers that were on leave for most of the time period tested. That is, the number of events belonging to these subject was irrelevant.
- They exhibited irregular behaviour patterns in that they did not have any of the significant traits observed for the regular workers.

These two unexpected groups were hence formed by subjects that did not have a large enough number of events associated with them and did unusual things (e.g. “shoulder surfing”). This is encouraging as it seems to indicate two things:

- High level of success may be attainable if low-support subjects are discounted from analysis, and
- (Small) groups may be formed for irregular behaviour pointing to potential outlier detection ability.

It is also interesting to note that there were other visitors with low support in the data. They, however, exhibited normal behaviour patterns and have been assigned to one of the three regular groups.

4.2 Complexity Issues

The resource requirements of the algorithmic solution described in this paper depend heavily on the design of how the event sequences are represented in memory, and how they are processed (added) as they are encountered in the evidence. As this representation is necessarily *complete* to facilitate further analysis, an upper bound to sequence length may need to be established to control memory requirements. This upper bound is dependent on two main factors: the number of subjects and the number of event types. When these two factors are fixed, the representation building algorithm is highly scalable due to its single pass nature and use of help pointers to insert/update new nodes in event sequences. Increasing the number of subjects increases memory and processing requirements linearly; increasing the number of event types, however, can cause an unexpectedly high increase in resource requirements as this increase is exponential to the length of sequence used (i.e. if the number of event types t increases by 1, the number of potential chains of length l increases from t^l to $(t + 1)^l$.) This may be alleviated by a more compact representation of event chains: by using a single representation for each distinct chain and providing look-up tables to determine the positional reference in the subject event chain tree hierarchy. This potential gain in memory requirements may, however, affect the efficiency of post-analysis by adding another layer of processing.

5 Conclusions

This paper presented a methodology to build and analyse event sequences from data acquired for computer forensic analysis. The approach introduces the concept of a subject for which sequences are generated. Analysing the frequency of these sequences and establishing sets of subjects based on shared sequences not only allows the individual description (profiling) of each subject but presents opportunities to contrast individual subjects for outlier analysis.

Some of the benefits of the algorithms described in this paper include the single pass design to access data and the building of an internal representation of what is being observed. This structure can then be directly used for further analysis without the need to consult external storage.

Another advantage is that frequent sequences are found while we are building the trees: once the ratio reaches a minimum frequency expressed globally (provided the number of events is known prior to execution) it will be frequent at the end - this is in contrast to some mining algorithms that can only determine this at the relevant iteration.

In the future, expanding the analytical capabilities of the methodology presented here would be desirable. In addition, finding the optimal maximum sequence length automatically while building the event chain representation may conserve memory requirements. Similarly, the on-the-fly trimming of unnecessary sequences from the representation could be enforced according to some pre-defined criteria. Finally, investigating alternate representations as mentioned in Section 4.2 may yield a more compact solution with a still acceptable performance to resource requirement ratio.

References

- Abbott, J. (2004), Writing log parsers for ECF, *Research Report*, Queensland University of Technology, 2004.
- Abraham, T. & de Vel, O. (2002), Investigative profiling with computer forensic data and association rules, In *Proceedings of the IEEE International Conference on Data Mining*, Maebashi City, Japan.
- Abraham, T., Kling, R. & de Vel, O. (2002), Investigative profile analysis with computer forensic log data using attribute generalisation, In *Proceedings of the Australasian Data Mining Workshop*, Canberra, Australia.
- Adomavicius, G. & Tuzhilin, A. (2001), Expert-driven validation of rule-based user models in personalization applications, *Data Mining and Knowledge Discovery*, 5(1/2):33–58.
- Adomavicius, G. & Tuzhilin, A. (2001), Using data mining methods to build customer profiles, *Computer*, 34(2):74–82.
- Aggarwal, C., Sun, Z. & Yu, P. (1998), Online algorithms for finding profile association rules, In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM-98)*, Bethesda, MD, USA.
- Agrawal, R., Imielinski, T. & Swami, A. (1993), Mining Associations between Sets of Items in Massive Databases, In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Washington D.C., pages 207–216.

- Agrawal, R. & Srikant, R. (1995), Mining sequential patterns, In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14.
- Casey, E. (2000), *Digital Evidence and Computer Crime*, Academic Press.
- Chan, P. K. (1999), A non-invasive learning approach to building web user profiles, In B. Masand & M. Spiliopoulou, editors, *Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*.
- de Vel, O., Anderson, A., Corney, M. & Mohay, G. (2001), Mining e-mail content for author identification forensics, *SIGMOD Record*, 30(4).
- Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, In *Proceedings of the Second Int. Conference on Knowledge Discovery and Data Mining*.
- T. Fawcett & Provost, F. (1997), Adaptive fraud detection, *Data Mining and Knowledge Discovery*, 1(3):291–316.
- Han, J., Cai, Y. & Cercone, N. (1992), Knowledge discovery in databases: an attribute-oriented approach, In *Proceedings of 18th Int. Conference on Very Large Databases*.
- Han, J. & Fu, Y. (1995), Discovery of multiple-level association rules from large databases, In *Proceedings of 21st VLDB Conference*.
- Hilderman R. J. & Hamilton, H. J. (1999), Heuristic measures of interestingness, In *Proceedings of the 3rd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'99)*.
- Hilderman R. J. & Hamilton, H. J. (1999), Knowledge discovery and interestingness measures: A survey, Technical Report CS-99-04, Dept of Computer Science, University of Regina.
- Hirsh, M., Basu, C. & Davidson, B. (2000), Learning to personalize, *Communications of the ACM*, 43(8):102–106.
- Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L. & Riedl, J. (1997), GroupLens: Applying collaborative filtering to usenet news, *Communications of the ACM*, 40(3):77–87.
- Lu, H., Feng, L. & Han, J. (2000), Beyond intra-transaction association analysis: mining multi-dimensional inter-transaction rules, *ACM Transactions on Information Systems*, 18(4):423–454.
- Mannila, H., Toivonen, H. & Verkamo, A. (1997), Discovery of Frequent Episodes in Event Sequences, *Data Mining and Knowledge Discovery*, 1(3):259–289.
- Mobasher, B., Dai, H., Luo, T., Sun, Y. & Wiltshire, J. (2000), Discovery of aggregate usage profiles for web personalization, In *Proceedings of the Workshop on Web Mining for E-Commerce (WEBKDD'00)*.
- Nanopoulos, A., Katsaros, D. & Manolopoulos, Y. (2001), Effective prediction of web-user accesses: a data mining approach, In *Proceedings of the Workshop on Mining Logdata Across All Customer Touchpoints (WEBKDD'01)*.
- Nesbitt, S. & de Vel, O. (1998), A collaborative filtering agent system for dynamic virtual communities on the web, In *Proceedings of the Conference on Learning and Discovery (CONALD98)*.
- Oates, T. & Cohen, P. R. (1996), Searching for structure in multiple streams of data, In *Proceedings of the Thirteenth International Conference on Machine Learning*.
- Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U. & Hsu, M. (2004), Mining sequential patterns by pattern-growth: The PrefixSpan approach, In *IEEE Trans. Knowledge and Data Engineering*, 16(10):1–17.
- Srikant, R. & Agrawal, R. (1995), Mining generalized association rules, In *Proceedings of 21st VLDB Conference*.
- Stanfill, C. & Waltz, D. (1986), Toward memory-based reasoning, *Communications of the ACM*, 29(12):1213–1228.
- Stevens, M. (2004), Unification of relative timeframes for digital forensics, *Digital Investigation*, 1(3):225–239.
- Tan, P. N. & Kumar, V. (2001), Mining indirect associations in web data, In *Proceedings of the Workshop on Mining Logdata Across All Customer Touchpoints (WEBKDD'01)*.