Activity Pattern Discovery from Network Captures

Alan C. Lin and Gilbert L. Peterson

***Abstract*— Investigating insider threat cases is challenging because activities are conducted with legitimate access that makes distinguishing malicious activities from normal activities difficult. To assist with identifying non-normal activities, we propose using two types of pattern discovery to identify a person's behavioral patterns in network data. The behavioral patterns serve to deemphasize normal behavior so that insider threat investigations can focus attention on potentially more relevant. Results from a controlled experiment demonstrate the highlighting of a suspicious event through the reduction of events belonging to discovered patterns.**

***Index Terms*—behavior, insider threat, pattern recognition**

I. Introduction

Insider threat involves malicious activity conducted with legitimate access [1], [2]. Legitimate access may be granted to an individual or impersonated by another entity [3], but because of the legitimate access, the nefarious activities are difficult to distinguish from normal actions until damages have occurred [4]. Upon discovering damages, an investigation then attempts to identify and analyze relevant fragments of computer data to piece together a probable explanation, or narrative, of events that transpired. This task is increasingly challenging because of the ever-increasing volume of digital data. Tools and technologies are necessary to efficiently triage data to find the relevant data items.

This paper proposes discovering patterns in user web activity to distinguish between accesses that conform to normal behavior from those that do not. The intent is that an investigation of an insider threat case can start with reviews of accesses that are inconsistent with a user's typical behavior patterns. The pattern discovery is an individualization process used to identify the anomalous events, during an investigation of an insider threat case. This is different from insider threat detection, which attempts to identify an insider threat actor from a set of actors [3]. The patterns from the pattern discovery are not used to uniquely identify the individual because the patterns may not be unique to only a single individual.

Web usage is contained in packet capture (PCAP) files. PCAP files contain low-level network frames of how the computer performs the underlying mechanics to make networked communication possible —determining the actions taken by the user that generated the frames takes technical expertise. To automate this task, this paper presents algorithms that extend alignment and bigram pattern discovery algorithms to identify behavior patterns in web usage data. Alignment pattern discovery leverages the Needleman-Wunsh [5] pairwise alignment algorithm from the bioinformatics domain, guided by a clustering algorithm that accounts for both content and edit distance similarity. The bigram pattern discovery uses a modified incremental activity recognition algorithm [6] that accepts patterns based on statistical expectation of occurrence. The two algorithms are complementary: the alignment pattern discovery aims to find patterns consistent across sessions while the bigram pattern discovery incrementally expands adjacent activities to find larger patterns.

II. Background

One area of research into insider threat detection leverages normal user baseline behaviors to perform anomaly detection. These approaches leverage the assumption that behavioral patterns are detectable from a person's behavior to build models of the users normal or abnormal behaviors.

A. Insider Threat Detection with Behavior Profiles

In Salem, et al.'s [3] survey of insider attack detection research, the authors identified web user profiling as a potential means to detect insider threat instances of stolen or borrowed credentials. The method used support vector machines the features of IP address, time of access, HTTP request method, and transfer size to detect instances of stolen credentials.

Liu, et al. [7] monitors system calls to leverage the benefit of complete monitoring and tamper-resistance. The authors attempt *n*-gram feature representation, citing successful application in external threat detection. However, they did not achieve similar results using *n*-gram features for insider threats. They attribute the difference in success due to the lack of change in application response from an insider using proper access rights, versus an outsider attack that does not.

Parveen and Thuraisngham [1] use unsupervised learning to

A. C. Lin and G. L. Peterson are both at the Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433 (e-mail: alan.lin@afit.edu and gilbert.peterson@afit.edu).

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

address unlabeled log data in the insider threat problem domain. Due to proper authorization in insider threat, log data is unlabeled because it does not readily show differences between normal and abnormal entries. Their work is similar to Lane and Brodley [8], which uses user UNIX command logs in anomaly detection. Both works assume that daily, consistent sequences are evidence of normal behaviors and use them to identify threats as significant excursions from the norm.

Legg, et al. [9] detect insiders through activity comparisons between the current daily observations, previously recorded observations, and observations from others in the same role. Their work used the CMU-CERT data, which included login, USB device, e-mail, web, and file access logs and synthetic scenarios that included similar fields. The observations are profiled in a tree-structure that is consistent between all users and roles to enable efficient comparison. Similarly, previous recorded behaviors are included in the "normal" profile, unless an attack was detected.

B. Behavior Profiles from Web Usage

Yang [10] created user profiles that describe repeating elements in a user's activity. The recurring nature of behavior was found in repeated visits to certain sites across sessions, repeated visits to the same site within a session, and the pattern of site visits. A session is defined as a continuous period of user web activity. Yang's support-based profiling technique provided the best results, where support was calculated as the number of sessions containing a discovered behavior pattern divided by the total number of sessions of a given user. Against a data set of 100 users, she found support-based profiling could achieve as high as 87% identification accuracy, if given a sufficient number user sessions and a large sliding window size to mine patterns. In cases with only a small sample of users, lift-based profiling, which is the frequency of a pattern within a user's session divided by the frequency of the pattern across all users, performed better.

Banse, et al. [11] performed behavior-based tracking using a Multinomial Naïve Bayes (MNB) classifier on Domain Name System (DNS) queries. An MNB classifier was selected for its computational complexity advantage over more advanced classifiers, such as support-vector-machines (SVM). Their goal was to identify the same user across multiple sessions in a data set that included 2100 users and where each user was represented by dynamic IP addresses, refreshed after a fixed amount of time. In cases where an IP address was classified as multiple users, the cosine similarity metric resolved the ambiguity. The combined MNB and cosine similarity model achieved as high as 88.2% identification, finding that user behaviors were stable provided sufficient data for a characteristic pattern to emerge.

McDowell [12] compared destination IP and DNS query methods of behavior modeling for anomaly detection. The data set was collected from a commuter military university to mimic traffic behavior from a government or corporate office. Their results using Naïve Bayes and K-Nearest-Neighbor classifiers found that DNS query performed better than destination IPs, but in general, did not achieve identification rates as high as either Yang [10] or Banse, et al [11].

The related area of research of anonymizers and realistic web traffic generation is motivated by evidence that web usage behavior is sufficient to identify a person. Banse, et al. [11] provided a disclaimer that some queries, despite using pseudonyms, still leak personally-identifiable information. Song, et al. [13] similarly discovered statistical fingerprints of behavior that reveal the host. The fingerprints are leveraged to classify individuals into groups, obfuscating the individual. The behavioral signature of the group is then used to generate representative network traffic without comprising personally identifiable information.

III. Methodology

The insider threat context for the proposed approach is where the forensic examiner is attempting to find the incriminating activities within the suspect's web usage activity. The overall process, shown in Fig. 1, begins with a preprocessing step that categorizes the sites visited to produce a linear sequence of events. The event sequences are necessary to perform the pattern discovery. The pattern discovery is an individualization process because the event patterns correspond to the individual's behavior. The data reduction step leverages this knowledge to deemphasize events that are attributable to the individual's typical behavior, leaving the unexplained events for further forensic examination.

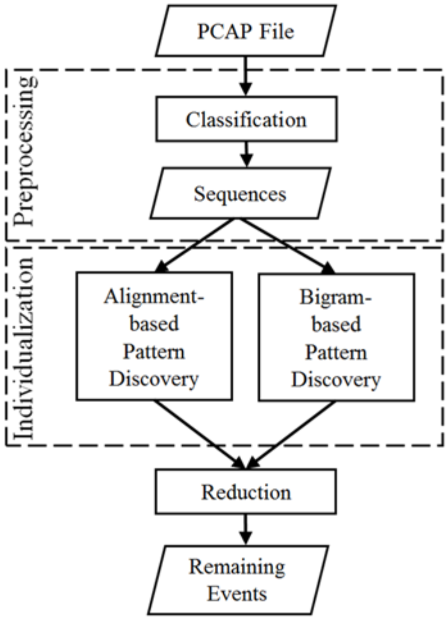

Fig. 1. Methodology Overview.

A. Pre-processing: Network data as a timeline sequence

The preprocessing step has several purposes. The first is to collapse multiple sequential visits to the same site. The effect of this collapse is that the ordering is the salient feature, rather than the duration of site visit. Site visit duration may be difficult to capture given the statelessness of web traffic. That is, it is difficult to know from just web traffic data whether a user left the terminal, or is conducting activities that do not generate network activity.

Second, site visits from secondary links are removed, such as those for advertisements. These web activities are based on the site's behavior and not actually reflective of the user's behavior.

Third, the preprocessing categorizes the sites as a reflection of the user goal or target. For example, visits to www.cnn.com and www.foxnews.com are both categorized as a `news` site visit. This can be done through organic keywords that reveal the category of websites or through a search of commercial databases, such as BlueCoat (www.bluecoat.com) or TrustedSource (www.trustedsource.org).

B. Pattern Discovery

The categories extracted from preprocessing become terminals in an activity sequence. From there, the activity sequences go through two kinds of pattern discovery to extract a user behavior pattern. The top-down approach uses *alignment-based pattern discovery* to identify overarching trends across sessions. Examples of overarching trends include Yang's example [10] where a person always starts a session by going to cnn.com. The bottom-up pattern discovery uses bigram-based pattern discovery, where activities that frequently occur together, in order, are considered patterns. In contrast to alignment, bigram pattern discovery starts locally and expands the patterns outward.

1) Alignment-based Pattern Discovery

The alignment-based pattern discovery uses the Needleman-Wunsh algorithm [5], [14] for pairwise alignment of two sequences. The Needleman-Wunsch algorithm, shown in Algorithm 1, requires a scoring system that rewards aligned symbols and penalizes gaps and mismatches.

A score matrix and a corresponding traceback matrix (lines 1 and 2) record the alignment path that determines aligned positions and insertions of necessary gaps. The algorithm has an *$O(mn)$* time and space complexity, where *m* and *n* are the length of the two sequences. Needleman-Wunsh progressively builds a multiple sequence alignment from a series of pairwise alignments as a performance consideration because simultaneous multiple sequence alignment algorithms incur an exponential computational complexity of *$O(2^k n^k)$*, where *k* is the number of sequences [15, Ch. 6]. The pairwise progression incrementally incorporates additional sequences to past alignments and back-propagates gaps into previous alignments when gaps are necessary to align the newest sequence. The trade-off of using pairwise alignment is that the overall alignment is sensitive to the order in which sequences are incrementally aligned due to the subsequently added gaps. To minimize gaps, a similarity matrix using clustering first identifies most similar sequences in a greedy approach based on a combined similarity measure as the arithmetic mean of content distance and edit distance. The reason why two kinds of similarity measures are used is to capture different aspects of similarity.

Algorithm 1: Needleman-Wunsh Alignment

```
Input: seq1 [1..m] : m-length string sequence
Input: seq2 [1..n] : n-length string sequence
Input: seq2 [1..n] : n-length string sequence
Input: init_penalty : initial misalignment penalty, ≤ 0
Input: gap_penalty : penalty for introducing gap, ≤ 0
Input: match_reward : reward for alignment, ≥ 0
Input: mismatch_penalty : penalty for non-alignment, ≤ 0
Output: align1 [1..x] : alignment of seq1 to seq2, x ≥ m
Output: align2 [1..y]: alignment of seq2 to seq1, y ≥ n
Output: alignment_score : higher score is greater alignment
 1 allocated score_matrix[m+1][n+1]
 2 allocate traceback_matrix[m+1][n+1]
 3 initialize alignment_score = 0
 4 for i = 1 to n
 5   score_matrix[i][0] = i * init_penalty
 6   traceback_matrix[i][0] = "up"
 7 end
 8 for j = 1 to m
 9   score_matrix[0][j] = j * init_penalty
10   traceback_matrix[0][j] = "left"
11 end

12 for i = 1 to n
13   for j = 1 to m
14     int s
15     if (seq1[j-1] == seq2[i-1] then s = match_reward
16     else s = mismatch_penalty
17     int diag = score_matrix[i-1][j-1]+s
18     int up = score_matrix[i-1][j]+gap_penalty
19     int left = score_matrix[i][j-1]+gap_penalty
20     score_matrix[i][j] = max(diag,up,left)
21     traceback_matrix[i][j] = max("diag", "up", "left")
22   end
23 end

24 i = n;  j = m
25 while ([i][j] != [0][0])
26   if (traceback_matrix[i][j] == "diag") then
27     align1.prepend(seq1[j-1])
28     align2.prepend(seq2[i-1])
29     i=i-1; j=j-1
30   else
31     if (traceback_matrix[i][j] == "left") then
32       align1.prepend(seq1[j-1])
33       align2.prepend("-")
34       j=j-1
35     Else
36       align1.prepend("-")
37       align2.prepent(seq2[i-1])
38       i=i-1
39     end
40   end
41 end
42 assignment_score = score_matrix[m+1][n+1]
```

Content similarity takes into account the bigram patterns, even if they are not aligned [6]. Content similarity is a combination of precision and recall, which are defined:

$$Precision = \frac{\left|\text{bigram}(v_1) \cap \text{bigram}(v_2)\right|}{\left|\text{bigram}(v_1)\right|} \tag{1}$$

$$Recall = \frac{\left|\text{bigram}(v_1) \cap \text{bigram}(v_2)\right|}{\left|\text{bigram}(v_2)\right|} \tag{2}$$

$$Content\ Similarity = 2 \times \frac{Precision \times \sqrt{Recall}}{Precision + \sqrt{Recall}} \tag{3}$$

The bigram(v) function returns the set of bigrams from the sequence, v, and the magnitude of the intersection that is in the numerator of equations (1) and (2) is the number of common bigrams, counting duplicates.

Edit distance takes into account similarity when there is alignment, even though bigram patterns are not preserved. To produce a result that can be used with content similarity, the edit distance is normalized by the length of the longer sentence. This way, both content and edit distance similarity is on a range between 0 (completely dissimilar) to 1 (identical).

$$\begin{gathered} Edit\ Distance\ Similarity = \\ \frac{\max\left(\text{length}\left(v_1\right), \text{length}\left(v_2\right)\right) - \text{edit distance}\left(v_1, v_2\right)}{\max\left(\text{length}\left(v_1\right), \text{length}\left(v_2\right)\right)} \end{gathered} \tag{4}$$

$$\begin{gathered} CombinedSimilarity = \\ ContentSimilarity + EditDistanceSimilarity \end{gathered} \tag{5}$$

Each cell in the similarity matrix is populated with similarity measures such as edit distance normalized to the longer sequence length. Then, the ordering begins with the two-most similar sequences and adds the remaining most similar sequence until the ordering includes every sequence, creating a guide-tree. Pair-wise alignment then uses the ordering to determine the incremental sequence of alignments.

2) Bigram-based Pattern Discovery

Bigram-based pattern discovery identifies patterns bottom-up, where activities that occur together in the same order form a pattern. The bigram-based pattern discovery algorithm, shown in Algorithm 2, is a modification of Peng, et al.'s [6] work in activity recognition. Two adjacent activities form a bigram, but a bigram is only accepted as a pattern if the bigram's joint-occurrence frequency is larger than the bigram's expected marginal frequency, tested using the chi-square test (line 27) on each bigram. Calculating chi-square in equation form shorthand is possible because there are only two items in the test.

Peng, et al.'s [6] attempt to identify similarities between bigrams in calculating significance. This was removed because these instances can be better captured under the alignment-based algorithm.

Another modification was made to the accept bigrams that occur exclusively together as significant. In these instances, line 27 will produce a divide-by-zero error. The implementation accepts these bigram if they occur over a user-defined frequency, despite not passing the chi-square test. This modification was done to better match the intent of the pattern discovery.

Algorithm 2: Bigram Pattern Discovery

Input: $W[1..n]$: list of n sequences
Input: m : merge threshold, ≥ 0
Output: $V_L[0..p]$: list of combined activities at level L
Output: W_L : W rewritten with V_L at each level

```
 1 initialize $L$=0
 2 initialize $R$ : map of bigrams (1st symbol, 2nd symbol)
 3 $V_0$ = terminals in $W$; $W_0 = W$;
 4 do
 5   $L$++
 6   ($W_L$, $V_L$) = collocation($W_{L-1}$,$V_{L-1}$,$L$,$R$)
 7 until $V_L$ is empty

 8 function collocation($W_{L-1}$,$V_{L-1}$,$L$,$R$)
 9   initialize jft : stores frequency of the bigram
10   initialize mft1: stores frequency of first symbols
11   initialize mft2: stores frequency of second symbols
12   foreach bigram in $W_{L-1}$
13     increment bigram count in jft
14     increment bigram's 1st symbol count in mft1
15     increment bigram's 2nd symbol count in mft2
16   end

17   foreach bigram in jft
18     $T$ = total number of bigrams in jft
19     $A$ = bigram count in jft
20     $E$ = bigram's 1st symbol count in mft1
21     $G$ = bigram's 2nd symbol count in mft2
22     $C = E - A$
23     $B = G - A$
24     $F = T - E$
25     $H = T - G$
26     $D = F - B$
27     $chi = T*((A*D)-(B*C))^2 / (G*H*E*F)$
28     if ($chi \geq m$)
29       add bigram to $V_L$
30       add bigram to $R$
31   end
32   rewrite $W_L$ in $V_L$
33 return ($W_L$,$V_L$)
```

3) Assumptions

We assume that the user does not interfere with the observation process and does not deliberately attempt to defeat the recognition system. Banse, et al. [11] identify some measures that complicate behavior-based tracking. Anonymizers like Tor obfuscate the destination IP address, therefore preventing classification of destination IP site. However, using anonymizers may be itself a suspicious behavior so detecting Tor traffic satisfies the goal of identifying suspicious behavior. Caching may also complicate sequence construction if they do not produce network traffic in the terminal assignment pre-processing step. Dynamic IP addresses are problematic if a single user is confused as multiple users because it decreases the sampling for behaviors. Range Queries, which hides user queries with random dummy queries, adds significant noise to pattern discovery.

The chi-square test will reject bigrams of symbols that exist only as a bigram because too many variables in the equation are zero. However, if the bigram frequently appears in joint frequency table, then rejecting the bigram produces counter-intuitive response based on the intent of pattern discovery. Thus, in addition to checking for divide by zero values, the chi-square test checks for situations for this occurs.

The implementation retains a history of bigram combinations at each level in V_L that makes the hierarchy evident. The highest level W_L define the S productions in the inferred grammar.

Bigrams are sensitive to the rewrite process. For instance, if `AB` and `BC` are both significant bigrams, `ABC` can be written as either `(AB)C` or `A(BC)`. Using alignment-based pattern discovery in concert mitigates this effect.

IV. Experiment

Greitzer and Ferryman [16] identified a lack of appropriate data and ground truth in developing and evaluating insider threat tools. As a pilot study, we captured a user's network traffic data over the course of three days. Truth data was recorded so that the actual user browsing sequence was known. Wireshark's [17] dumpcap utility was used to capture the packets.

A synthetic insider threat inject scenario asked the user to visit www.HSBC.com during the middle of the second session. This site was selected because it is not suspicious site based on the URL (TrustedSource categorized the link as minimal risk with a `banking` web category). This was done to mirror the fact that no explicit policy should exclude the user from performing this action and that the user has legitimate privilege to perform this action. However, the act of visiting this banking website interrupts the user's normal habits on when they typically carry out banking tasks with shopping activities. The HSBC visit is also not uniquely identifiable based on visit frequency, as there are other sites only visited once.

In this regard, this scenario considers both the normal user behavior, as well as the malicious behavior to be detected [4]. Similar to the scenarios generated in [9], using a banking site as the anomalous observations fits an insider threat narrative where the actor is confirming payment after completing some kind of malicious activity. The injection testing approach, similarly used in [16] and [18], also addresses the privacy concerns of the captured material while ensuring presence of known insider threat data.

Two runs were conducted on this scenario to examine sensitivity to the preprocessing step that generates the activity sequences. The first run uses a set of five activities while the alternate run increased the activity set by one to better capture the webmail URL. In both cases, the injected insider threat activity survives the reduction process for further investigation.

A. Initial Run with Five Activity Terminals

To examine reduction based on individualization in pattern discovery, the process deemphasizes events attributable to a discovered pattern. With the injected scenario, the site visit to www.HSBC.com should not be deemphasized.

Fig. 2 shows the activities from the three sessions, t1, t2, and t3, consisting of 41 total activities. The red highlights the scenario injected event.

The sessions do not include activities from ad services, typically indicated by the referred-from field in the stream because they are not reflective of user-driven activities. The sessions also do not include activities caused by background services such as antivirus updates or operating system updates for the same reason.

The pre-processing step uses the IP address metadata and organic keywords retrieved from ipaddress.com as well as McAfee's Threat Intelligence database at www.trustedsource.org to classify the different activities into terminals. Fig. 3 shows the sequences of events consisting of five categories: `edu` (education), `socnet` (social networking), `news`, `shopping`, and `banking`.

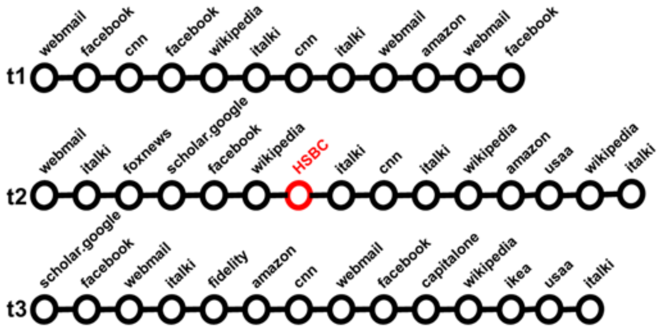

Fig. 2. User activities from three sessions: t1, t2, t3. The red highlights the scenario injected event that reduction process should not eliminate.

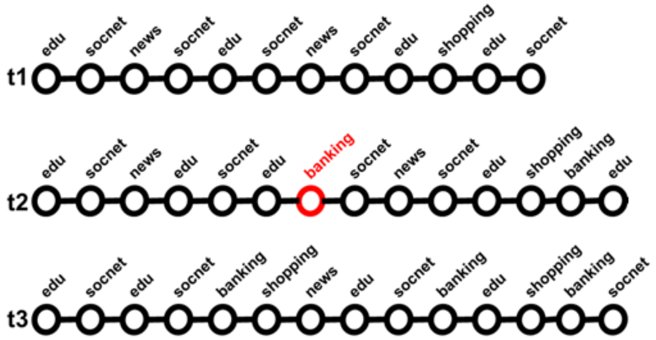

Fig. 3. Sessions rewritten by categories.

The alignment-based inference process identified activity patterns that occur across the all of the sessions, shown in Fig. 4. The events denoted in blue are those that occur similarly across t1, t2 and t3. The gray events denote breaks between aligned events.

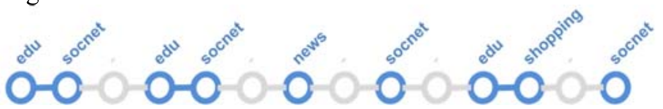

Fig. 4. Alignment of the three sessions. The blue events correspond to alignments. The gray dashes correspond to breaks between aligned events.

Events in the sessions that correspond to the blue events in the alignment are grayed out in Fig. 5 to show that they are deemphasized.

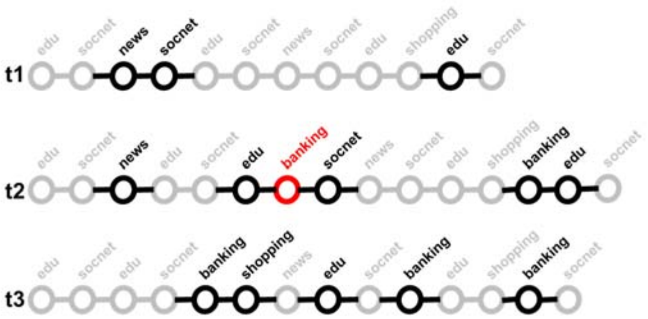

Fig. 5. Sessions with aligned symbols deemphasized.

Bigram pattern discovery identifies frequent activity patterns, focusing on increasingly longer patterns of adjacent symbols. Deemphasizing these patterns from the sequences in addition to those deemphasized by the alignment-based pattern discovery further reduces the number of remaining activities. Bigram pattern discovery identified the following patterns, where nested-bigrams are chunked with parenthesis:

```
shoppingbanking,
socnetnews
edusocnet
socnetedu
newssocnet
(edusocnet)news
(shoppingbanking)socnet
 edu(shoppingbanking)
(socnetedu)(shoppingbanking)
(socnetedu)shopping
(edusocnet)banking
```

The vocabulary terms signify additional behavioral patterns. The parentheses indicate a previously merged bigram within another bigram. Activities that are unexplained by the alignment are matched against the vocabulary list. Sequences that appear in the vocabulary list are also de-prioritized, shown in washed-out green in Fig. 6. The bigram discovery process is independent of the alignment inference process. Therefore, the results from the bigram process can reduce the event sequences on their own. By using both approaches, activities can be explained away using both methods. An activity exclude through alignment can still be used as part of a bigram to exclude activities not explained by the alignment discovery process. To highlight these occurrences, activities as part of bigrams that were deemphasized in the alignment step are relabeled green, but retain the gray circle.

In comparing results between the two processes, the alignment included two adjacent activities, edusocnet, that was also discovered in the bigram inference approach. The alignment also included another adjacent pair, edu and shopping, which did not appear on the bigram vocabulary list. However, edushopping appears in the vocabulary list three times, under edu(shoppingbanking), (socnetedu)(shoppingbanking) and (socnetedu)shopping, marking it as part of other frequent patterns.

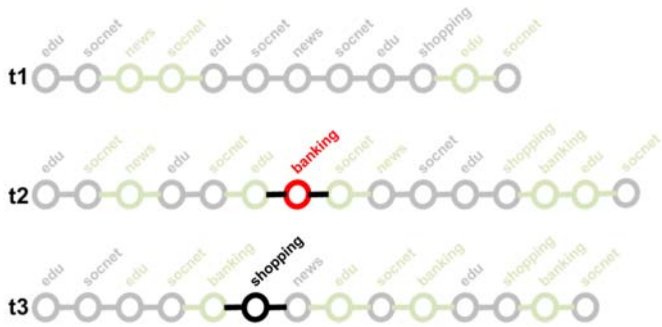

Fig. 6. Sessions after deemphasizing events from discovered alignment and bigram patterns.

At the point of investigation in Fig. 6, only two activities remain emphasized across the three sessions that are unexplained by discovered behavior patterns. These two activities indicate where to begin the investigation, which includes the artificially injected event.

B. *Alternate Run with a Sixth Activity Terminal*

The initial run uses five activity categories in the activity sequences. This run adds an additional activity terminal to investigate the sensitivity of the pattern discovery algorithms on the preprocessing step.

The URL for the webmail activity had an ".edu" extension and was assigned the edu terminal based on the ipaddress.com classification. We repeat the individualization process and obtain a reduction that categorized webmail activity with a comm terminal, as a better reflection of the actual activity. This expanded the category size to six. As expected, increasing the variety of symbols increases the distance between sequences and this change produced two clusters: (t1, t2) and (t3). An increase in the number of clusters means that there is less likely to be a single alignment pattern, because the alignment pattern from one cluster does not transfer to other clusters. This is illustrated in Fig. 7, where t3 does not have any activity in gray; all washed out activities in t3 are due to bigram patterns.

The resulting change is a decrease in the amount of activities explainable as part of a pattern, leaving five activities unattributed. Most importantly though, the injected insider threat activity, denoted in red, remains in the activity set for further investigation, even with the additional terminal.

The introduction of the `comm` terminal also changed the bigram vocabulary:

```
shoppingbanking
commsocnet
socnetnews
(commsocnet)banking
(commsocnet)news
socnetedu
((commsocnet)banking)shopping
socnet(commsocnet)banking
edusocnet
```

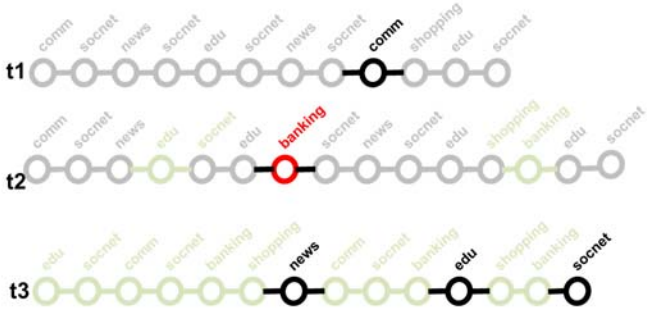

Fig. 7. Sessions using six categories. Gray denotes de-emphasis from alignments. Green denotes de-emphasis from bigrams. The black and red are remaining events from reduction.

C. Discussion

The motivation for using web usage data is based on successes in earlier works [10]–[12]. An advantage of the proposed approach is not needing to define a sliding window parameter. The alignment-based pattern discovery compares entire sessions and the bigram-based pattern discovery iteratively expands a window size until no patterns are found. The other advantage of the proposed approach is the preprocessing that aggregates websites of the same type, reducing sensitivity to dynamic IP assignments which may treat them as different sites.

With regards to disadvantages, the proposed process assumes that attacks are preceded by observable events to indicate future malicious activity [19] in addition to the supposition that most malicious behavior is anomalous. These assumptions, however, are still subject to debate within the insider threat research community. The impact of these assumptions is that if a suspect does not have a routine behavior pattern, then the pattern discovery would not find patterns to inform the reduction process and fail to eliminate activities for examination.

In addition, web traffic alone may not be sufficient to fully reconstruct a criminal timeline. For example, the Digital Forensics Research Conference 2008 scenario [20] included both packet captures and a memory dump, both of which were essential to attributing culpability of the crime. This disadvantage is shared across all insider threat approaches that only examine one set of features. To address this issue, Eldardiry, et al. [18] and Legg, et al. [9] propose multi-domain approaches to insider threat detection. The pattern discovery methods proposed in this paper, are extensible to incorporate multiple domains. In the activity recognition domain, Peng, et al. [6] created a hierarchy to organize multiple sensor readings, before conducting the pattern discovery to mine patterns that include activities across sensors. Similarly, a hierarchy can organize usage activity across multiple domains, such as file access or log-in activity, though the additional domains must also employ pre-processing to produce sequences of discrete symbols.

V. Conclusion

Insider threat is challenging to address because the insider attack uses legitimate access, making it difficult to separate normal activities from those that are not. This paper proposes using pattern discovery of network data to identify an individual's behavior patterns. With known behavior profiles, an investigator on an insider threat case can focus on activities that the actor had authority to perform, but does not conform to past behavior. The forensic examiner can then focus on a reduced set of user activities in the investigation. Results on a controlled lab data set retained the injected anomalous activity through the reduction process.

For future work, the pattern discovery section relies on preprocessing to generate activity sequences. Testing this process on a larger scale will require automating the preprocessing step, such as those in [21], [22]. The preprocessing will also need to overcome additional issues such as networks that use caching, which may require different classification techniques.

Additionally, pattern discovery also requires further investigation to determine the set of categories that accurately represent web activity, yet still produce activity patterns. Identifying this set requires a data collection of multiple users over a longer period.

Finally, this approach may be useful in generating data that mimics usage behaviors, but does not compromise the privacy of the users in the data collection. To produce research data of a user's usage behavior, a generator uses the discovered patterns to create an artificial web usage log. The generator reverses the pre-processing step by selecting a site from the category, a generator can choose from a list sanitized sites that do not leak personally identifiable information from the source data collection.

References

[1] P. Parveen and B. Thuraisingham, "Unsupervised incremental sequence learning for insider threat detection," *2012 IEEE Int. Conf. Intell. Secur. Informatics*, pp. 141–143, 2012.

[2] A. Azaria, A. Richardson, S. Kraus, and V. S. Subrahmanian, "Behavioral Analysis of Insider Threat: A Survey and Bootstrapped Prediction in Imbalanced Data," *IEEE Trans. Comput. Soc. Syst.*, vol. 1, no. 2, pp. 135–155, 2014.

[3] M. Ben Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," *Insid. Attack Cyber Secur. Beyond Hacker*, pp. 69–90, 2008.

[4] R. Mills, M. Grimaila, G. Peterson, and J. Butts, "A scenario-based approach to mitigating the insider threat," *ISSA Journal*, pp. 12–19,

2011.

[5] V. Likic, “The Needleman-Wunsch algorithm for sequence alignment,” *Lect. given 7th Melb. Bioinforma. …*, pp. 1–46, 2008.

[6] H. K. Peng, P. Wu, J. Zhu, and J. Y. Zhang, “Helix: Unsupervised grammar induction for structured activity recognition,” *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 1194–1199, 2011.

[7] A. Liu, C. Martin, T. Hetherington, and S. Matzner, “A comparison of system call feature representations for insider threat detection,” *Proc. from 6th Annu. IEEE Syst. Man Cybern. Inf. Assur. Work. SMC 2005*, vol. 2005, pp. 340–347, 2005.

[8] T. Lane and C. E. Brodley, “Sequence matching and learning in anomaly detection for computer security,” in *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, 1997, pp. 43–49.

[9] P. a Legg, O. Buckley, M. Goldsmith, and S. Creese, “Automated Insider Threat Detection System Using User and Role-Based Profile Assessment,” *IEEE Syst. J.*, pp. 1–10, 2015.

[10] Y. Yang, “Web user behavioral profiling for user identification,” *Decis. Support Syst.*, vol. 49, no. 3, pp. 261–271, 2010.

[11] C. Banse, D. Herrmann, and H. Federrath, “Tracking users on the Internet with behavioral patterns: Evaluation of its practical feasibility,” *IFIP Adv. Inf. Commun. Technol.*, vol. 376 AICT, pp. 235–248, 2012.

[12] C. M. McDowell, “Creating Profiles From User Network Behavior,” Naval Postgraduate School, 2013.

[13] Y. Song, S. J. Stolfo, and T. Jebara, “Behavior-based network traffic synthesis,” in *Technologies for Homeland Security (HST), 2011 IEEE International Conference on*, 2011, pp. 338–344.

[14] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins.,” *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, 1970.

[15] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*, vol. 101, no. 474. 2004.

[16] F. L. Greitzer and T. A. Ferryman, “Methods and metrics for evaluating analytic insider threat tools,” *Proc. - IEEE CS Secur. Priv. Work. SPW 2013*, pp. 90–97, 2013.

[17] G. Combs, “Wireshark.” 2015.

[18] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka, “Multi-Domain Information Fusion for Insider Threat Detection,” *2013 IEEE Secur. Priv. Work.*, pp. 45–51, 2013.

[19] P. Legg, “Guest Editorial : Emerging Trends in Research for Insider Threat Detection Guest Editorial : Emerging Trends in Research for Insider Threat Detection,” *JoWUA*, vol. 5, no. 2, pp. 1–6, 2014.

[20] “DFRWS 2008 Forensics Challenge Challenge Data and Submission Details,” 2008. [Online]. Available: http://www.dfrws.org/2008/challenge/submission.shtml. [Accessed: 18-Aug-2014].

[21] G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, and Y. Jin, “ReSurf: Reconstructing Web-Surfing Activity From Network Traffic,” *IFIP Netw. Conf.*, pp. 1–9, 2013.

[22] C. Neasbitt, R. Perdisci, K. Li, and T. Nelms, “ClickMiner : Towards Forensic Reconstruction of User-Browser Interactions from Network Traces Categories and Subject Descriptors,” *Proc. 2014 ACM SIGSAC Conf. Comput. Commun. Secur. ACM*, pp. 1244–1255, 2014.