2016 IEEE Security and Privacy Workshops

# LINEBACKER: LINE-speed Bio-inspired Analysis and Characterization for Event Recognition

Christopher S. Oehmen, Paul J. Bruillard, Brett D. Matzke, Aaron R. Phillips, Keith T. Star, Jeffrey L. Jensen, Doug Nordwall, Seth Thompson, Elena S. Peterson
Pacific Northwest National Laboratory
Richland, WA USA

***Abstract*—The cyber world is a complex domain, with digital systems mediating a wide spectrum of human and machine behaviors. While this is enabling a revolution in the way humans interact with each other and data, it also is exposing previously unreachable infrastructure to a worldwide set of actors. Existing solutions for intrusion detection and prevention that are signature-focused typically seek to detect anomalous and/or malicious activity for the sake of preventing or mitigating negative impacts. But a growing interest in behavior-based detection is driving new forms of analysis that move the emphasis from static indicators (*e.g.* rule-based alarms or tripwires) to behavioral indicators that accommodate a wider contextual perspective. Similar to cyber systems, biosystems have always existed in resource-constrained hostile environments where behaviors are tuned by context. So we look to biosystems as an inspiration for addressing behavior-based cyber challenges. In this paper, we introduce LINEBACKER, a behavior-model based approach to recognizing anomalous events in network traffic and present the design of this approach of bio-inspired and statistical models working in tandem to produce individualized alerting for a collection of systems. Preliminary results of these models operating on historic data are presented along with a plugin to support real-world cyber operations.**

***Keywords— biosequence model, cybersecurity, leaky buckets;***

## I. INTRODUCTION AND RELATED WORK

Network traffic facilitates a wide range of intended uses across a globally distributed set of endpoints. This traffic embodies *behaviors* that span the needs of users and systems. As a result, network traffic contains attributes that can form the basis of models of behavior that can be used to distinguish various usage types. The driving force behind line speed bio – inspired analysis and characterization for event recognition (LINEBACKER) is the assertion that behavior models can help distinguish user-consistent behavior from inconsistent behavior (anomalies) such as communicating with malicious websites, large data exfiltration, or a number of other attacks. Ultimately, to be of practical value, behavior models should support decision-making by humans or automated processes.

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency (HSARPA), Cyber Security Division (DHS S&T/HSARPA/CSD), via Interagency agreement, number HSHQPM-12-X-00097, and by the Signature Discovery Initiative, Laboratory Directed Research and Development program at Pacific Northwest National Laboratory, operated for the DOE by Battelle under contract DE-AC06-76RLO-1830.

There is a great deal of prior research in behavior-based network intrusion prevention and detection. For some reviews, see [1-5]. Some of these methods are probabilistic in nature [6], while others are built on data mining and fusion techniques [7, 8]. In such a data-rich environment, machine learning approaches have also been shown to hold great promise [9-13], in some cases leveraging multi-processor environments to accommodate the needs of large scale data analysis [14]. In addition to conventional desktop IT systems, behavioral approaches have been proposed for cyber physical systems, [15] and wireless systems [16, 17]. LINEBACKER has been introduced at a high level in a prior paper [18], but in the current work the authors present much more detail about how the method is implemented, and include case studies of how it is being used operationally.

In the present work, the emphasis is on *sequence-based* models of behavior. Because many aspects of computers and complex networks have sequential features, sequence models have been used to model cyber behaviors. For example, packet train models have been used to optimize network architectural features [19]. Document access sequence models have been used to optimize caching of web content to enhance speed of access [20]. Many applications in anomaly detection rely on sequence models [21]. LINEBACKER employs two specific sequence models—*sequence alignment,* which is drawn from biological sciences, and *leaky buckets*, which was originally designed for network traffic control.

*Sequence alignment* is a general-purpose technique for finding an optimal alignment between strings of text given a reward/penalty system [22, 23]. BLAST is a particular implementation of sequence alignment, tuned for biosequences, specifically protein sequences and DNA sequences [24, 25]. BLAST focuses on finding *local alignments,* or substrings that align, rather than *global alignments*, in which entire strings are forced to align. ScalaBLAST is a high performance implementation of BLAST that has been used previously to enable processing of very large datasets such as are commonly found in multi-genome scale biology and cybersecurity [26, 27]. More recently, BLAST has been refactored to be used on data sets for applications outside biology [28] and the feasibility of this


DOI 10.1109/SPW.2016.44

method for non-biological cybersecurity applications has been demonstrated [29, 30].

The *leaky bucket algorithm* has been used as a method for preventing congestion in networks with variable sequences of utilization such as ATM [31] and packet switch networks [32]. Leaky buckets use a virtual bucket having fixed "volume", allowing requests or traffic to accumulate in a reserve queue, and drain this queue at a constant rate. The constant drain modulates the flow of traffic from the bucket, preventing downstream congestion without losing data until the bucket overflows. This method was adapted in the current work as a behavior model by learning the bucket volume and flow rate for bursty traffic from a collection of systems. Each system's volume and rate constitute a simplistic behavior model, and alarms can be raised when a user's bucket overflows, or runs empty for a given behavior type. This has a damping effect on the otherwise highly variable distribution of traffic, while still allowing for a high-level characterization of the overall behavior.

Motivated by prior work in sequence models for other applications, the authors investigated the suitability of sequence-based approaches for use in network traffic analysis. This investigation focused on three key research questions:

- At what level of granularity of network traffic is a sequence-based model most useful?
- Is there a collection of basic types of network traffic events that has a determinstic, quantitative mapping from raw or processed traffic records that preserves the ability to recognize anomalous events? If so, how many features are needed to resolve the difference between these basic types?
- What aspects of the biosequence representation can be used to construct models that capture meaningful behaviors in network traffic data?

This paper reports on the results of research with emphasis on three novel contributions. First, network traffic flow (which is summary information about the data exchanged between systems) proved to be the most promising level of granularity for developing sequence-based models. Second, the space of network flows for the datasets used in this study can be characterized by fewer than 8 principal components, though in the general case this is expected to depend heavily on the location of monitoring and what type of filtering is used. As a result, there is a natural mapping from network flow records to a manageable number of labeled clusters that allow for network traffic to be represented as sequences and analyzed using biosequence analysis. Third, it was observed that 1) systems typically exhibit a small number of behavior types, leading to biosequence representations having low complexity; and 2) the volume of flows associated with systems was bursty, but these bursts still had characteristics on which models of anomalous behavior could be based. So behavior models in LINEBACKER were developed using a combination of two features: *complexity* of biosequences to detect changes in the types of network flows associated with systems, and *leaky buckets* to detect changes in the bursty properties of each system's network traffic. This set of models was operationalized and examples of novel detections enabled by them are presented.

The remaining sections in this manuscript are organized as follows: Section II describes the data used and methods for constructing the biosequence complexity and leaky buckets models. Section III contains the results of applying these models to synthetic and live data. Section IV provides some case studies including examples of unusual traffic that was detected using LINEBACKER. Section V is a discussion on aspects of LINEBACKER performance. Section VI concludes the manuscript with a summary of contributions and impact of the work.

## II. METHODS

The key for LINEBACKER models is converting network flow traffic to a sequence of individual events. For the biosequence model, this conversion process makes it possible to represent the raw traffic in a way that can be clustered where each cluster can be assigned a letter. After learning this labelling for network traffic, the cluster labels on arbitrary network data are used to convert each traffic record into a character of text by finding its nearest labeled cluster. A sequence of labels then becomes the character sequence that is used in biosequence analysis (see Figure 1). The raw vector form can also be used for additional behavior models such as leaky bucket analysis.

### A. Converting network traffic to behavior models

This approach utilizes network flows, which contain descriptions of the attributes of communications between systems. Flow data contains IP addresses, ports, and protocols for the source and destination systems, as well as data about the number of exchanges and amount of data passed between the systems but not the actual information being exchanged. Specifically, the method presented here uses *bidirectional* flows, which resolve multiple one-way communications between systems into longer "sessions" of communications that can go in either direction between the systems.

To map network flow records to characters of text, bidirectional flow data was converted to numerical vectors of fixed length describing each flow session where each feature was determined from one of the fields available from standard network flow tools. First, the field indicating which transport protocol was being used served as a filter to restrict the analysis to TCP protocol, but the method could be applied to other protocols as well. Of the remaining standard bidirectional flow fields, the following were used as features in the fixed length vector representation of a network session: duration, number of bytes sent from source, number of bytes sent from destination, number of packets from source, and number of packets from destination. For each of these fields, the value from the records was augmented by 1, and the log10 value was used to prevent vastly different counts from creating ill-conditioned datasets. Additionally, the source and destination ports were represented in the feature vectors by a "1" if the

respective port was one of {21, 22, 25, 80, 443, 445}, which are the commonly allowed ports for TCP traffic. The port fields were assigned a value of "0" if the respective port was not a member of the list. A final vector element was included for source and destination ports that had a value of "1" if the respective port was larger than 1024, and a value of "0" otherwise. Ports above 1024 are registered and often used by particular applications. Though other data are available that could be used to determine the vector form of network flows, the data chosen for this vector form is commonly available from network collectors so that LINEBACKER would be most widely deployable in operational environments.

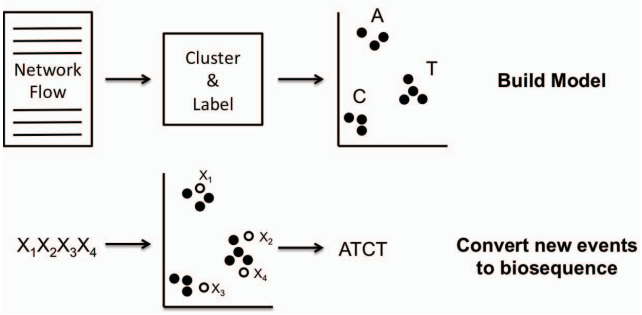


**Figure 1. Process for converting network flow traffic to sequences for analysis**

Characterizing the space of session records was the next step. Multiple clustering techniques were applied to this data. K-means was selected because the clusters were found to be fairly robust with respect to the clustering method and parameters chosen (results not shown). K-means clusters were built on vector representations of bidirectional flow records for a large collection of data (roughly 6 million records), which revealed that there are optimally 7 clusters present. For representing network flows in vector form, this means only 7 unique characters are needed in the alphabet used to describe the sequence of network flows emanating from a single system or between a pair of systems. Figure 2 illustrates the justification for the assertion that on the data under study, this low number of characters forms a sufficient alphabet. In Figure 2, the number of clusters in K-means clustering is varied and the measure of variance within each cluster (the "withinss" value") is plotted. The "withinss" value should drop when increasing the number of clusters if using this higher number results in centroids that truly represent distinct Gaussian clusters. At some point, the value of "withinss" ceases to drop as a function of adding clusters, suggesting that the additional clusters no longer capture true divisions in the data. As a result, the "elbow" of such a plot is a good measure of the optimal number of clusters.

Principal component analysis (PCA) was also performed on this data as another method to determine the number of sub behaviors that could be used to convert raw records to letters of text for biosequence analysis. Supporting the K-means clustering results, PCA confirmed that 8 principal components accounts for 96.8% of the variance in the data (results not shown).

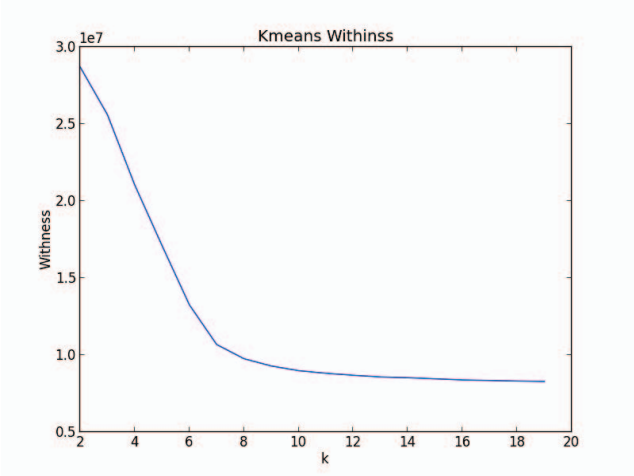


**Figure 2. Analysis of clustering on network traffic**

*B. Biosequence Complexity Model*

As noted above, network flow data was translated into a vector form, making it possible to cluster a collection of traffic records. The above analysis suggests the proper number of clusters for the data is 7. So 7 clusters were generated from a historical dataset and their centroids were each labeled with a single character. Biosequences of network traffic were generated from those letters by assigning the letter of the nearest centroid for each flow record in a sequence of network traffic sessions for a given source system. Sequences could be generated that represent the behavior of the originating IP address for a period of time, or that of the destination IP address, or of the conversation between 2 IP addresses.

Biosequences generated from a large collection of systems revealed that each system exhibits remarkably low complexity. Though there were 7 labeled clusters and hence 7 characters in the sequence alphabet, most systems were highly biased to exhibiting only 2 or 3 characters in particular distributions. This led to the use of *sequence complexity*, as opposed to the actual sequence, as the basis for the model.

Biosequence complexity models are an essential component of sequence analysis tools such as BLAST, which must limit the search space over which to explore text alignments using an underlying statistical model. Masking low complexity subsequences is a key step in BLAST because low complexity sequence regions violate the assumptions of its underlying statistical model. SEG [33] is a complexity model for proteins having a base alphabet size of 20 characters, again with a few additional characters used to capture uncertainty in the character sequence. Improvements in accuracy over SEG for larger alphabets are possible by introducing more complex techniques such as machine learning [34], but this degree of computing is unnecessary for our purposes, so for the current work, a simplified version of the SEG algorithm was adapted for use in LINEBACKER.

This sequence complexity model is a sliding window of 50 characters in length over the biosequence representation of network flow sequences. For each window, the complexity is calculated as the entropy, expressed in terms of a discrete set of background probabilities $p_i$ of seeing each character of the alphabet so that entropy $H$ as a function of characters $x_i$ over alphabet size $n$ is given by the standard information theory definition [35]. The size of the sliding window was determined through previous experience with these types of sequences and some trial and error (method not reported).

Alarms are generated from the complexity model output by calculating the deviation for a given window, defined as the absolute value of the difference between the maximum value of window-entropy and the average from previous sequences on the same system. For a given time of the sequence, the overall deviation score is the maximum deviation score across all window deviation scores, and is selected and associated with the beginning of the sequence.

### C. *Leaky Buckets Model*

Leaky buckets are a technique for controlling flow of bursty events, such as are common in cyber datasets. They control flow by establishing a "bucket" to queue the events, and a constant "leak" rate. When the bucket is not empty, the leak rate will allow the events to leave the bucket queue at a constant rate, draining the bucket. There are two key tuning steps in developing a leaky bucket model: 1) giving each bucket a finite capacity that is reasonable for each system, and 2) choosing the rate of leak from each bucket. The first mechanism allows the bucket to overflow in abnormal situations where too many of the same event are being collected in a given bucket. The second mechanism makes it possible to quantify whether network events filling the bucket are increasing or decreasing without the need for a statistically anomalous event to occur.

For LINEBACKER, the attributes of bucket volume and leak rate were used to tune each system with a behavior model of its own. For each system, these parameters were trained from prior behavior, capturing essential past performance features of each system. Using a decaying sliding window, each model was continually updated to accommodate newly observed behaviors. To apply this model to real systems, models were created by creating two buckets for each system of interest, one for bytes of data sent from the system as a source, and one for bytes of data sent to the system as a destination. For each system, mean and standard deviation source and destination byte counts were calculated using the traffic from training periods. The mean byte count was used for each system's buckets as the leak rate. Then for each system, the same traffic was played back into the bucket and means and standard deviations in bucket "fill level" were calculated. The mean and standard deviation of all non-zero data entering each bucket and maximum amount (maximum burst) entering a bucket per unit time were also calculated. Bucket capacity for each model was assigned to be maximum burst size plus the mean bucket level plus 6 standard deviations of the bucket level plus six standard deviations of the burst size. This ensured that no bucket would overflow on the first day, and that for a bucket to overflow on any subsequent day, the address's behavior would have to be significantly different than that from the first day.

## III. MODEL DEPLOYMENT

Though exhaustive validation has not yet been done on these models, early results are reported here suggesting that the models can be used to detect anomalous behaviors and that the alert output provided by a security information and event management (SIEM) solution could streamline the processes of getting new behavior models into an operational setting.

LINEBACKER is currently deployed in an operational environment to support evaluation of the interface and behavior models by cyber defenders. Approved bidirectional network flow data is captured on a machine running the LINEBACKER tool suite at a rate of ~25 million flows every 5 minutes. A script checks for incoming network flow approximately every 15 minutes, and passes the network flow records to both the complexity and leaky bucket models. For the sequence complexity model, the data is first passed through a converter that applies labeled cluster nearest distance measures to the new data and produces biosequence-formatted representation for each system. For the leaky bucket model, the raw data is passed to a python implementation of the leaky bucket algorithm. Deviations in the sequence complexity models are detected by calculating rolling averages and standard deviations for each system against these averages. The model code writes time stamped alarms for each violating system to log files when a predefined number of standard deviations is exceeded for that system with respect to the rolling baseline model. These log files are forwarded to the cyber defenders' SIEM, are parsed, and presented in graphical format to the analyst. This flow of data within the collection and analysis system is illustrated in Figure 3.

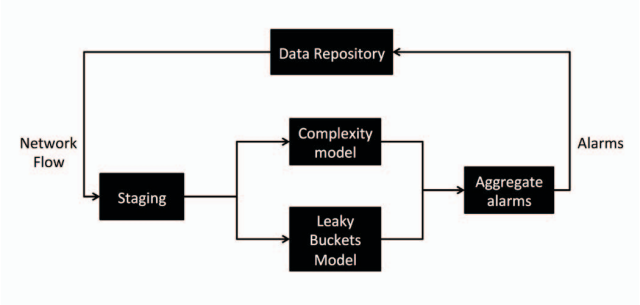


**Figure 3. LINEBACKER analysis pipeline**

A dashboard has been developed and configured to monitor the behavior model log files in real time. As log information is written to the log files, the SIEM indexes the data and the dashboard view is updated. There are five provided panels based on queries to the indexed log data, as illustrated in Figure 4. Each bar refers to the model output from a single system. So the collection of bar charts is a collection of output from the worst offending systems over the collection and analysis time window.

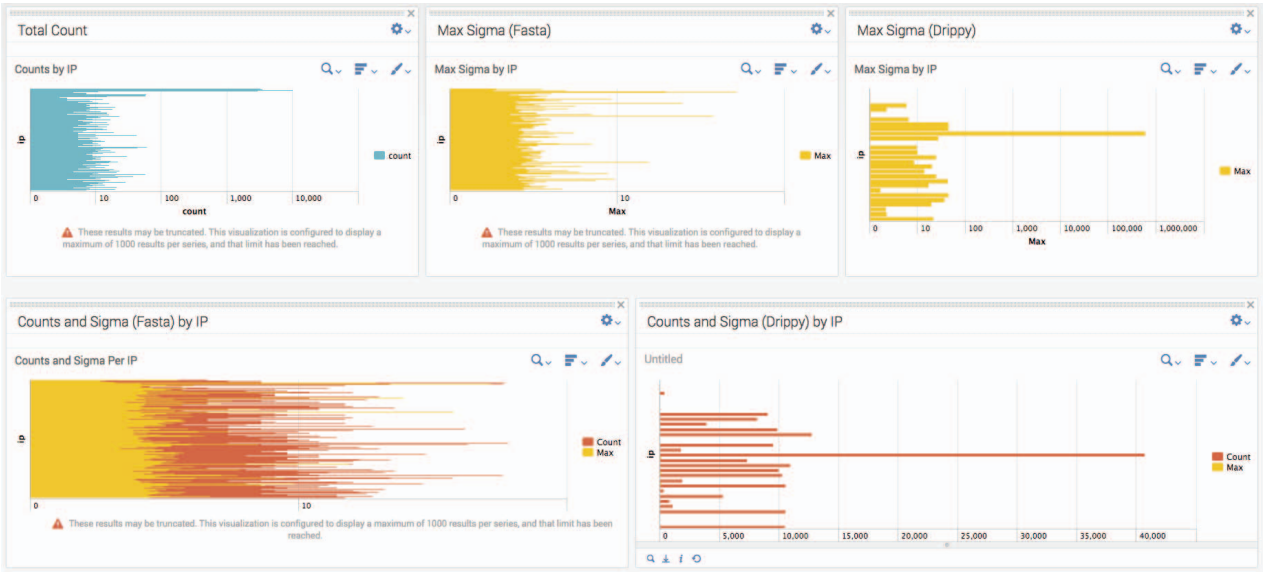


**Figure 4. Dashboard view of LINEBACKER showing the most anomalous IP addresses according to the biosequence and leaky buckets models.**

From the top left in Figure 4 and progressing left to right: Panel 1 (total count) shows the number of times an alarm has occurred for each of the top offending systems for both the sequence complexity model and leaky bucket model. The length of the line indicates the number of times the alarm was detected. From this view, it is easy for defenders to recognize highly anomalous systems. Panel 2 (max sigma FASTA) shows the highest alarm value seen for each of the top offending systems for the sequence complexity model. In this case the length of the line indicates severity of the worst deviation in over a fixed time window. The time window over which this maximum deviation is recorded can be varied. Similar to the "total count" panel, this view highlights systems that are highly anomalous, but in terms of the severity of their anomaly, not the number of anomalous events detected. Panel 3 (max sigma DRIPPY) is similar to the previous panel, but it shows the highest alarm value seen for each of the top offending systems for the leaky bucket model over the configurable time window. Panel 4 (counts and sigma FASTA by IP) overlays panels 1 and 2, and shows the number of times an alarm has occurred and the maximum observed value of the alarm for each of the top offending systems for the sequence complexity model. Panel 5 (counts and sigma DRIPPY by IP) combines panels 1 and 3, showing the number of times an alarm has occurred and the maximum observed value of the alarm for each of the top offending systems for the leaky bucket model.

## IV. Case Studies

Since LINEBACKER has been deployed on live data there have been several instances of anomalous activity that have been identified. This includes examples where LINEBACKER assessments were corroborated using more conventional tools. It also includes examples where LINEBACKER was the only tool to detect an anomaly. In some cases the anomalies were associated with malicious behavior, and in some cases the anomalies were just unusual events, but the cybersecurity defenders were glad to be alerted to the strange behavior. A collection of these anecdotal examples is included here to underscore the potential value of using these models to augment existing anomaly detection tools.

### *A. LINEBACKER on a known bad event*

To evaluate the LINEBACKER models on real network data, a collection of network flow records was obtained for a system on a live network that was found to be communicating with a known "bad" IP address (from a current blacklist). Results for the sequence complexity model and leaky bucket models on this data are illustrated in Figure 5. Because the sequence complexity model is event-based and the leaky bucket model is time-based, events are not aligned on their horizontal axes. However, correlating the time points in the model records revealed that both models alarmed around the time of known communication of the system with a known bad IP. It is interesting to note that both models detected anomalies on the system, but these detections have different features. The biosequence model showed greater response during phase 1, and the leaky bucket model showed a greater response during phase 2.

### *B. Events that were not detected using conventional tools*

The biosequence model on its own detects some events not recognized using conventional tools or the leaky buckets model. A single day of operational data was selected at

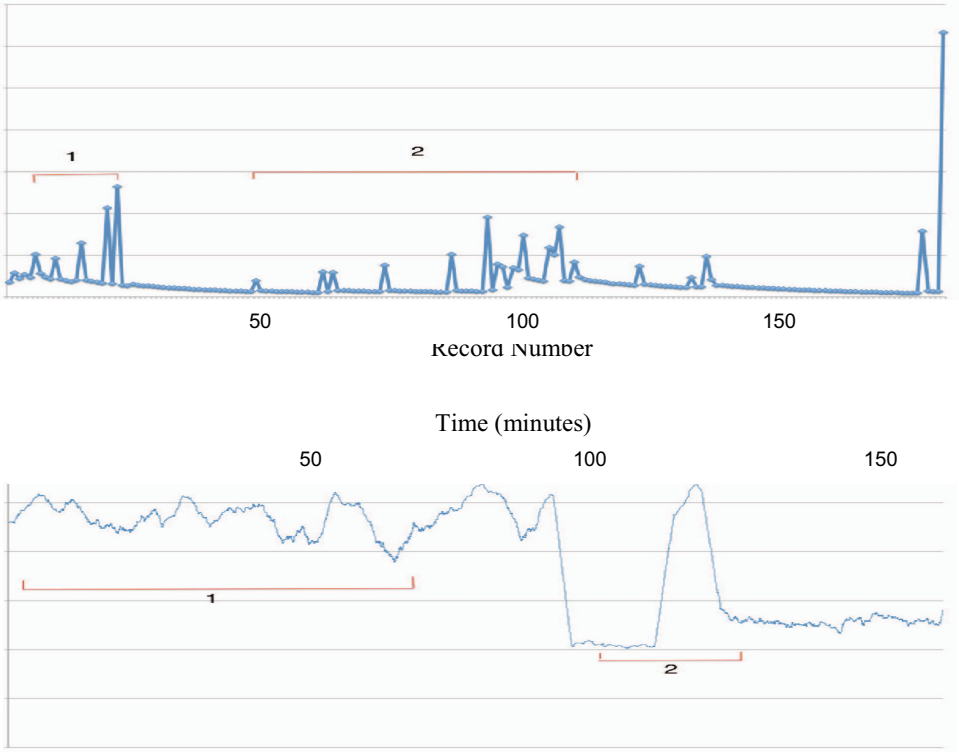


**Figure 5. LINEBACKER behavior models for sequence complexity (top) and leaky bucket (bottom) for a system on a live network involved in communications with a known bad IP address over the span of several days.**

random and reviewed by a member of our operations staff. By only looking at the top offenders in the biosequence model logs, the analyst found a system that had been infected that none of the other tools identified and which did not rise to the top of the leaky bucket model alert list. This system had large spikes in web traffic to an IP address that has been associated with malware and malicious domains.

Similarly, the leaky bucket algorithm on its own sometimes identifies anomalous systems that are missed by conventional tools and the biosequence model. As an example, in the same full day dataset used in the prior example, the most anomalous system identified using the leaky bucket model was determined to be a true positive. In this case, the anomalous system was the target of a failed attack. Many of the incoming packets were blocked and associated with many other alerts (blocked TCP and ICMP scans, NTP attacks, RPC, and MS-SQL exploits.) This would be an ideal system to monitor for future suspicious activity and alert at a lower threshold.

In some cases, finding systems that have lower alerts, but that appear anomalous from both model perspectives is useful. For the same day of operational data in the prior two examples, six systems raised alarms in both the biosequence and leaky bucket models. One of these systems was found to have been associated with a "strange" file transfer behavior that turned out to be benign. A second address was found to be involved in a vulnerability scan. Neither of these events was detected by existing tools and both were considered anomalous. Taken together, these case studies illustrate the value of combining multiple behavior models to detect cyber anomalies.

## V. Discussion and Future Directions

### A. *Additional models*

The LINEBACKER system could be used with additional behavior models than those presented in this work. The current version has two relatively simplistic models implemented, but the combination of them is potentially more meaningful than either model on its own. As a result, in addition to being an operational tool, LINEBACKER could be a powerful research tool to allow for rapid deployment of new models that produce summarized alert output in a format that is readily usable by network defenders and decision makers. In future work the authors anticipate engaging a wide community of behavior model researchers to expand the functionality of LINEBACKER and to provide a means to test and evaluate new models.

### B. *Scalability*

The deployed LINEBACKER system was benchmarked with respect to its ability to keep up with large enterprises. The current implementation has demonstrated the ability to convert network traffic records into the biosequence model description at a rate correlating to dozens of national laboratory-sized sites simultaneously. Running the *models* on this dataset has only been benchmarked at the scale of a single site's raw data (~5

million flows/minute), but no bottlenecks in performance have yet been observed at this scale. In future work the scalability of LINEBACKER models applied to larger system sizes will be explored.

More extensive validation will also be performed using a combination of publically available test datasets and live data from deployed instances of LINEBACKER to further test the performance of the tool and its behavior models.

### C. Behavior models to support cross-site data sharing

One of the benefits of using behavior models as a means to capture, analyze, and describe network activity is that model results themselves can be shared between sites without divulging the raw data that comprised the model. In effect, models become an intermediate language that could be communicated between sites that do not wish to share or are legally or otherwise compelled to not share raw data. In this view, each site already has its own data collection permissions, policy, and tools. LINEBACKER operating at each site would convert site-specific network flow data into model results that indicate types of anomalous behavior. Anomalies detected from the behavior models could be directly communicated between sites without the need for exchanging raw data or other site-specific details. Additionally, LINEBACKER could function as an enabler for neutral information sharing and analysis centers (ISACs), such as those created for each of the U.S. critical infrastructure areas. In this view, data from participating partners could be operated on by LINEBACKER either at the endpoints, or after collection by the ISAC. Alerts could be expressed in terms of model deviations and shared without reference to the specific underlying data that led to the alert. Sites having instances of the anomalous alerting behavior could carry out their own responses and would have access to their own detailed data for further situational awareness and support for decision-making.

### D. Applying sequence analysis to other network data

In prior network-specific work, the use of biosequence analysis techniques has been demonstrated on analysis of *full packets* which is the complete content of web pages and other network traffic, and *URL's*, which are the text strings used in web browsers to locate web sites [36]. URLs tend to be relatively short and the matching subsequences may be detectable using more simplistic regular expressions. Additionally, the lexical content of URLs is inconsistent. Some URLs have natural language expressions in them, and some are gibberish when read by a human, so interpreting similarities in URLs is difficult. For full packets, the sequence similarity method was able to detect pages that have similar content in the same order, even when some content was not identical. But the value of finding similar information content is not compelling for cyber defenders because the information being exchanged is too low-level in many cases to associate it with malicious or benign behaviors or intent.

LINEBACKER results presented in this paper are all based on models built using network flow data. This was found to be the optimal level for these models to operate because it is an intermediate level of granularity between the very coarse level of URL analysis and the very fine-grained level of full packet analysis. This allows LINEBACKER to not be overly focused on the details of what information is exchanged between computers, and simultaneously not so general as to miss the more complex behavioral indicators, as is the risk with URL analysis.

## VI. Conclusions

This study focused on exploring how biosequence and other models could be used to capture essential features of network traffic for the purpose of identifying anomalies. To that end, three key contributions are presented herein.

First, after exploring the use of biosequence analysis at various levels including URL's, network flows, and full packets, it was observed that network flow had enough content to form the basis of a model without being too voluminous or detailed, as full packets are. This allowed the team to explore specific implementations of biosequence and leaky buckets models on network flow data.

The second major contribution of this work is the finding that network traffic represented as bidirectional network flows is conducive to being mapped to character sequences using an alphabet of seven characters. This captures most of the essential variance in the flows and utilizes a natural partition of the network flow characteristics into a small number of dimensions. Translating sequences of network traffic into a biosequence representation allowed for exploration of system behaviors using traditional biosequence analysis tools with the goal of identifying common subsequences, or *motifs* in biological parlance, that are indicators or health or compromise.

Using these methods revealed a very low complexity behavior for most systems. While this is typically not useful in biological analysis, it provided the basis for the second bio-inspired aspect that was developed—that of sequence complexity as the basis for a behavioral model, which is the third main contribution of this study. The discovery of low complexity in the system sequences was leveraged to construct system-specific models that can differentiate between normal and abnormal behavior. Taken from the same roots as the low complexity masking algorithms in traditional biosequence analysis, this complexity model gives a novel view into system behavior using insight gained from bio-inspired analysis of the cyber systems.

When combined with a more traditional network traffic model and applied to live traffic, the resulting analysis detected examples of verified anomalous behavior, corroborated by existing tools in some cases, and in some cases discovering otherwise undetected anomalous behavior.

## Acknowledgment

The authors would like to thank PNNL operations management and cyber defense teams for their willingness to evaluate LINEBACKER on live data and provide examples for case studies.

## REFERENCES


[1] Anderson, E. and J. Choobineh, *Enterprise information security strategies.* Computers and Security, 2008. **27**: p. 22-29.

[2] Sperotto, G., et al., *An Overview of IP Flow-Based Intrusion Detection.* IEEE Communications Surveys & Tutorials, 2010. **12**(3): p. 343-356.

[3] Garcia-Teodoro, P., et al., *Anomaly-based network intrusion detection: Techniques, systems and challenges.* Computers and Security, 2009. **28**: p. 18-28.

[4] Modi, C., et al., *A survey of intrusion detection techniques in the cloud.* J. Network and Computer Applications, 2013. **36**: p. 42-57.

[5] Zhou, C., C. Leckie, and S. Karunasakera, *A survey of coordinated attacks and collaborative intrusion detection.* Computers and Security, 2010. **29**: p. 124-140.

[6] Su, C.-C., et al., *The new intrusion prevention and detection approachs for clustering-based sensor networks*, in *Wireless Communications and Networking Conference 2005*. 2005. p. 1927-1932.

[7] Bass, T., *Intrusion Detection Systems and Multisensor Data Fusion.* Communications of the ACM, 2000. **43**(4): p. 99-105.

[8] Ertoz, L., et al., *MINDS-minnesota intrusion detectoin system*, in *Next-Generation Data Mining*. 2004. p. 199-218.

[9] Bivens, A., et al., *Network-Based Intrusion Detection Using Neural Networks*, in *Intelligent Engineering Systems through Artificial Neural Networks (ANNIE-2002)*. 2002, ASME Press: St. Louis, MO. p. 579-584.

[10] Deng, H., Q.-A. Zeng, and D. Agrawal, *SVM-based Intrustion Detection System for Wireless Ad Hoc Networks*, in *Vehicular Technology Conference, 2003. VTC 2003-Fall.* 2003. p. 2147-2151.

[11] Li, W., *Using Genetic Algorithm for Network Intrusion Detection*, in *Proceedings of the United States Department of Energy Cyber Security Group*. 2004. p. 1-8.

[12] Zhang, J., M. Zulkernine, and A. Haque, *Random-Forests-Based Network Intrusion Detection Systems.* IEEE Trans. Systems, Man, and Cybernetics-- Part C: Applications and Reviews, 2008. **38**(5): p. 649-659.

[13] Zhang, J. and M. Zulkernine, *Anomaly Based Nework Intrusion Detection with Unsupervised Outlier Detection*, in *IEEE International Conference on Communications, 2006, ICC'06*. 2006. p. 2388-2393.

[14] Paxson, V., R. Sommer, and N. Weaver, *An Architecture for Exploiting Multi-Core Processors to Parallelize Network Intrusion Prevention*, in *IEEE Sarnoff Symposium, 2007*. 2007. p. 1-7.

[15] Cheung, S., et al., *Using Model-based Intrusion Detection for SCADA Networks*, in *SCADA Security Scientific Symposium*. 2007. p. 1-12.

[16] Mishra, A., K. Nadkarni, and A. Patcha, *Intrusion Detection in Wireless Ad Hoc Networks.* IEEE Wireless Communications, 2004: p. 48-60.

[17] Zhang, Y. and W. Lee, *Intrusion Detection in Wireless Ad-Hoc Networks*, in *6th Annual International Conference on Mobile Computing and Networking, (MOBICOM) 2000*. 2000, ACM: Boston, MA. p. 275-283.

[18] Oehmen, C., E. Peterson, and B. Cox, *Behavior Models to Express and Share Threat Information.* IT Professional, 2015. **17**(5): p. 12-14.

[19] Jain R and S. Routhier, *Packet Trains-Measurements and a New Model for Computer Network Traffic* IEEE J. on Selectec Areas in Communications, 1986. **SAC-4**(6): p. 986-995.

[20] Yang, Q., H. Zhang, and T. Li, *Mining Web Logs for Prediction Models in WWW Caching and Prefetching*, in *KDD '01 Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 2001, ACM: San Francisco, CA. p. 473-478.

[21] Chandola, V., A. Banerjee, and V. Kumar, *Anomaly Detection for Discrete Sequences: A Survey.* IEEE Trans. Knowledge and Data Engineering, 2012. **24**(5): p. 823-839.

[22] Needleman, S. and C. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins.* J. Mol. Biol., 1970. **48**(3): p. 443-453.

[23] Smith, T., M. Waterman, and C. Burks, *The statistical distribution of nucleic acid similarities.* Nucleic Acids Res., 1985. **13**(2): p. 645-656.

[24] Altschul, S., et al., *Basic local alignment search tool.* J. Mol. Biol., 1990. **215**(3): p. 403-410.

[25] Altschul, S., et al., *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.* Nucleic Acids Res., 1997. **25**(17): p. 3389-3402.

[26] Oehmen, C. and D. Baxter, *ScalaBLAST 2.0: Rapid and Robust BLAST Calculations on Multiprocessor Systems.* Bioinformatics, 2013. **29**(6): p. 797-798.

[27] Oehmen, C. and J. Nieplocha, *ScalaBLAST: A scalable implementation of BLAST for high-performance data-intensive bioinformatics analysis.* Trans. Parallel Distributed Sys., 2006. **17**(8): p. 740-749.

[28] Peterson, E., et al., *Novel Visual and Analytical Methods in Repurposing Legacy Scientific Code- A Case Study*, in *2013 International Conference on Software Engineering Research and Practice*. 2013.

[29] Oehmen, C., E. Peterson, and S. Dowson, *An organic model for detecting cyber events*, in *Proc. Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, F. Sheldon, et al., Editors. 2010, ACM: Oak Ridge, TN.

[30] Oehmen, C., E. Peterson, and J. Teuton, *Evolutionary Drift Models for Moving Target Defense*, in *Eighth Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW '13)*, F. Sheldon, et al., Editors. 2013, ACM: Oak Ridge, TN. p. Article No. 37.

[31] Butto, M., E. Cavallero, and A. Tonietti, *Effectiveness of the "Leaky Bucket" Policing Mechanism in ATM Networks.* IEEE J. Selected Areas in Communications, 1991. **9**(3): p. 335-342.

[32] de Veciana, G., *Leaky Buckets and Optimal Self-tuning Rate Control*, in *Global Telecommunications Conference, 1994*. 1994, IEEE. p. 1207-1211.

[33] Wooten, C. and S. Federhen, *Statistic of local complexity in amino acid sequences and sequence databases.* Computers and Chemistry, 1993. **17**(2): p. 149-163.

[34] Barber, C. and C. Oehmen, *An Efficient Machine Learning Approach to Low-Complexity Filtering in Biological Sequences*, in *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 2012: San Diego, CA. p. 237-243.

[35] Shannon, C. and W. Waver, *The Mathematical Theory of Communiation*. 1963, Urbana, IL: University of Illinois Press.

[36] Teuton, J., et al., *LINEBACkER: Bio-inspired Data Reduction Toward Real Time Network Traffic Anomaly Detection*, in *1st International Symposium on Resilient Cyber Systems*. 2013: San Francisco, CA.