Improving Intrusion Detection Performance Using Keyword Selection and Neural Networks

Richard P. Lippmann
MIT Lincoln Laboratory, Rm S4-121
244 Wood Street
Lexington, MA 02173-0073
rpl@sst.ll.mit.edu
phone: (781) 981-2711

Robert K. Cunningham
MIT Lincoln Laboratory, Rm S4-129
244 Wood Street
Lexington, MA 02173-0073
rkc@sst.ll.mit.edu

Abstract

The most common computer intrusion detection systems detect signatures of known attacks by searching for attack-specific keywords in network traffic. Many of these systems suffer from high false-alarm rates (often 100's of false alarms per day) and poor detection of new attacks. Poor performance can be improved using a combination of discriminative training and generic keywords. Generic keywords are selected to detect attack preparations, the actual break-in, and actions after the break-in. Discriminative training weights keyword counts to discriminate between the few attack sessions where keywords are known to occur and the many normal sessions where keywords may occur in other contexts. This approach was used to improve the baseline keyword intrusion detection system used to detect user-to-root attacks in the 1998 DARPA Intrusion Detection Evaluation. It reduced the false alarm rate by two orders of magnitude (to roughly 1 false alarm per day) and increased the detection rate to roughly 80%. The improved keyword system detects new as well as old attacks in this data base and has roughly the same computation requirements as the original baseline system. Both generic keywords and discriminant training were required to obtain this large performance improvement.

1. Introduction

Heavy reliance on the internet and worldwide connectivity has greatly increased the potential damage that can be inflicted by remote attacks launched over the internet. It is difficult to prevent such attacks by security policies, firewalls, or other mechanisms because system and application software always contains unknown weaknesses or bugs, and because complex, often unforeseen, interactions between software components and/or network protocols are continually exploited by attackers. Intrusion detection systems are designed to detect attacks which inevitably occur despite security precautions.

A review of the many alternative approaches to intrusion detection is available in [1]. The most common approach to intrusion detection, often called "signature verification," detects previously seen, known, attacks by looking for an invariant signature left by these attacks. This signature may be found either in host-based audit records on a victim machine or in the stream of network packets sent to and from a victim and captured by a "sniffer" which stores all important packets for on-line or future examination. The Network Security Monitor (NSM) was an early signature-based intrusion detection system that found attacks by searching for keywords in network traffic captured using a sniffer. Early versions of the NSM [2] were the foundation for many government and commercial intrusion detection systems including NetRanger [3] and NID [4]. This type of system is popular because one sniffer can

monitor traffic to many workstations and the computation required to reconstruct network sessions and search for keywords is not excessive. In practice, these systems can have high false-alarm rates (e.g. 100's of false alarms per day) because it is often difficult to select keywords by hand which successfully detect real attacks while not creating false alarms for normal traffic. In addition, these signature-based systems must be updated frequently to detect new attacks as they are discovered.

Two research systems [5,6] that took part in the 1998 DARPA off-line intrusion detection evaluation [7,8] provided good performance for user-to-root attacks where local users on a UNIX host illegally obtain root-level privileges. Their performance was much better than that of an untuned baseline keyword reference system which had a detection rate of only 20% at a false alarm rate above 100 false alarms per day. The purpose of the research described in this paper was to determine whether the simple baseline keyword system can be modified to obtain similar performance improvements and to analyze those factors that contribute to improved performance. Techniques which are successful in improving the baseline system could also be used to improve existing commercial and government keyword-based systems. Two of the most important factors which were explored were adding new generic keywords to detect actions associated with attack components and using discriminant neural network training. This work focused on analysis of network traffic obtained using a sniffer, on attacks aimed at UNIX hosts, and on attacks where a local user illegally obtains root privileges on a victim machine.

Table 1: Attack types in test data.

	Solaris	SunOS	Linux
Old	eject ffbconfig fdformat	loadmodule	perl
New	ps	ps	xterm

2. DARPA 1998 Intrusion Detection Data Base

This research was made possible by the availability of a large-scale realistic intrusion detection data base created under DARPA funding in 1998 [7,8]. More than two months of network traffic were generated that was similar to the type of traffic observed flowing between U.S. Government sites and the internet. Traffic and attacks were generated on a network which simulated 1000's of Unix hosts and 100's of users. Attacks were launched from outside the simulated site against Linux, SunOS, and Solaris Unix victim hosts on the inside. Sniffing data containing all bytes transmitted to and from this simulated site were used for system development and testing along with attack labels and timing information. The seven weeks of training data provided in this data base were used for training and the two weeks of test data were used for testing.

Seven types of user-to-root attacks shown in Table 1 were included in this data base. Five of these were considered "old" attacks that were provided in training data and two were "new" attacks visible only in test data. These attacks employ many mechanisms to illegally achieve root-level privileges including buffer overflows, system misconfigurations, and race conditions. A subset of attacks were made stealthy both by concealing the text of attack shell, perl, and C source code exploit scripts and by spreading the attack across multiple telnet sessions. All attacks included telnet sessions to the victim machines. Seven weeks of training data contained roughly 1,200 telnet sessions and 34 attack instances while two weeks of test data contained roughly 12,900 telnet sessions and 35 attack instances (most of the 12,900 test telnet sessions are components of denial of service and password guessing attacks). Only training data was used for system development. Formal evaluation guidelines described in [9] were followed for testing, and only one pass through the test data was performed for the final evaluation. Results in this paper were not obtained as part of the official DARPA 1998 evaluation. They are unofficial results not directly comparable to results obtained for systems in the official evaluation.

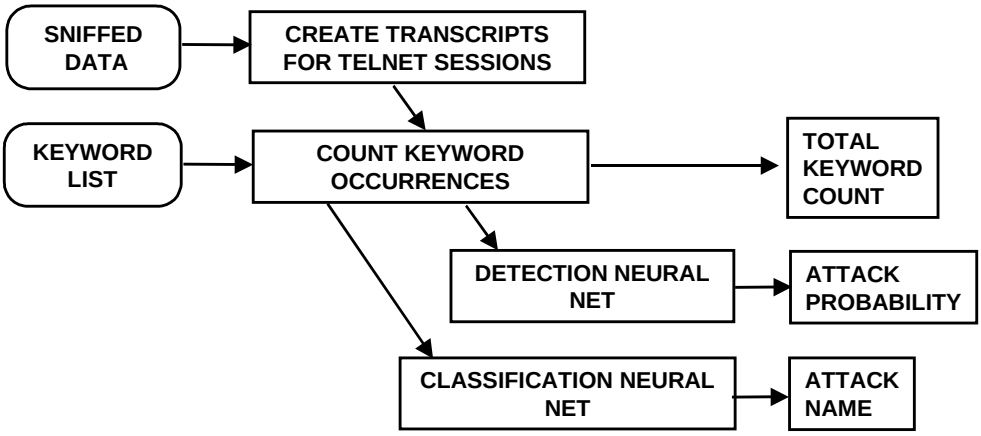

Figure 1.Block diagram of the baseline and enhanced intrusion detection systems.

3. Improving the Baseline Keyword System

Figure 1 shows a block diagram of the improved keyword system with discriminant training and of the reference intrusion detection system. Network sniffing data is first processed to reconstruct transcripts containing all bytes transmitted to and from victim hosts during telnet sessions. The characters in each telnet session that were returned from the destination to the source are then processed to produce counts of the number of times each keyword, on a pre-defined list, occurs in each session. The total number of keyword counts (across all keywords) is the first output. This count is used as a reference. It is similar to the keyword count score produced by the untuned keyword baseline reference system that was part of 1998 DARPA evaluation [8]. Its score is also similar to the session warning score produced by many keyword-based intrusion detection systems. Ideally, this count would be monotonically related to the probability of an attack in a telnet session. In the improved system, keyword counts are further processed by neural networks. One network weights keyword counts to provide an improved estimate of the posterior probability of an attack in each telnet session and a second network attempts to classify known attacks and thus provide an attack name.

All pattern classification experiments were performed using LNKnet pattern classification software [10]. Initial experiments were performed with 58 keywords selected to be representative of those keywords used in existing keyword-based intrusion detection systems at the time of the 1998 DARPA evaluation. These include keywords that detect suspicious actions (e.g. “passwd”, “shadow”, “permission denied”, “+ +”) and keywords that detect well- known attacks (e.g. “from: |”, “login: guest”). This keyword list was then expanded based on a hand examination of attacks in the training data. It was found that many attacks included attack code downloading, attack preparation, the actual break-in where a new root shell was often created, and actions performed after root-level privilege was obtained. Keyword selection first involved looking for words or word strings that might occur primarily during attacks. Keywords were added to detect downloading attack code (e.g. “uudecode”, “>ftp get”), setup actions (e.g. “chmod”, “gcc”), a new root shell (e.g. “root:”, “# ”), a debug statement that some buffer overflow code prints when a root shell is successfully created (“Jumping to address”), the clear-text version of attack exploit source code (e.g “fdformat”, “ffbconfig”), and some actions performed after the break-in (e.g. “linsniff”, “.rhost”). In addition, strings were added to detect the operating system of victims to help classify attacks (e.g. “SunOS UNIX”, “Red Hat Linux”). A total of 31 new keywords were added to the existing 58 old keywords. An attempt was made to select generic keywords that would generalize across attacks and also keywords that would be difficult to hide. For example, responses from system programs were used as keywords instead of input strings typed by users. In addition, strings used by programs such as uuencode to delimit text transfers were used as keywords instead of the commands

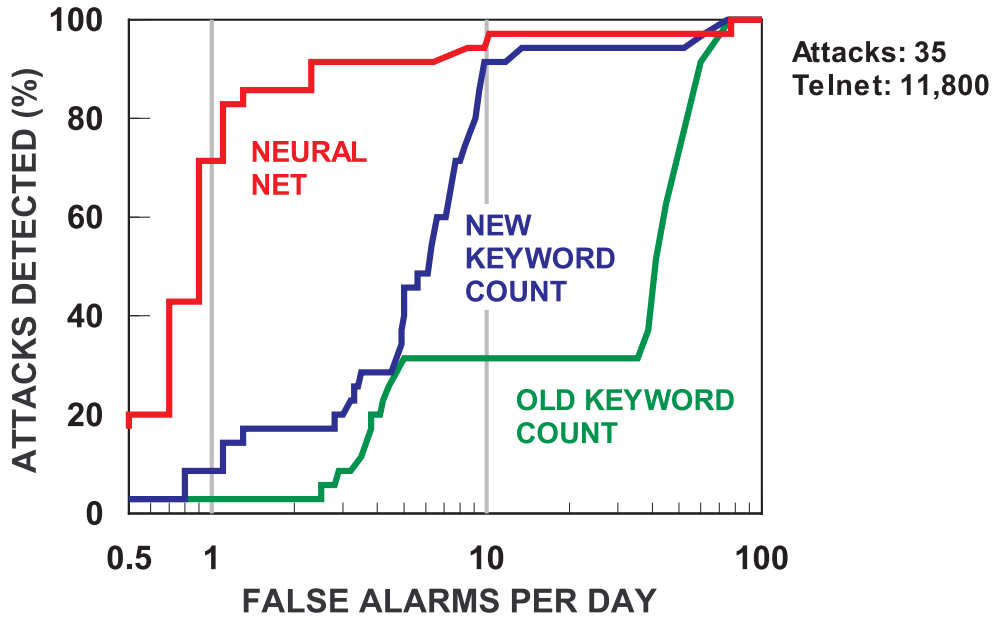

Figure 2. Receiver Operating Characteristic (ROC) curves for a reference system using the total keyword count based on 58 old keywords, for a second reference system using the total keyword count based on 89 old and new keywords, and for a neural network system using 30 old and new keywords

that set up these transfers. Regular expression string matching was used to anchor keywords to the beginning of lines, allow an arbitrary number of spaces or tabs between multiple words, and allow arbitrary digits in some strings.

Ten-fold cross-validation experiments were performed using multi-layer perceptron classifiers and the training data to select both keywords and the network topology for the detection neural net. Most experiments were performed using stochastic gradient descent learning, a squared-error cost function, 20 epochs of training, and a step size of 0.01. Keywords were selected using both weight magnitude pruning for networks with no hidden nodes, and forward-search feature selection. Good detection performance and a low false- alarm rate was obtained using 30 keywords, a single-layer network with no hidden units, and a sigmoid output unit. The false-alarm rate increased with 35 or more keywords and the detection rate decreased with 25 or fewer keywords. The most important ten keywords with strongly positive weights associated with attacks were all new keywords designed to detect transmission of attack scripts to target machines (e.g. “cat >”, “uudecode < ”), a new root shell (“uid=0(root)”, “bash# ”), and post-attack actions or components of attack scripts (e.g. “linsniff”, “ffbconfig”). Some keywords selected to detect attacks were given strong negative weights (e.g. “# ”, “root”, “/etc/motd”) which indicates that these keywords are more common in normal sessions than in attack sessions. This result demonstrates the importance of discriminant training and of testing keywords on normal sessions to determine their ability to detect attacks while generating few false alarms. Selecting keywords solely on the basis of attack sessions may lead to excessive false alarms because keywords that appear to be essential to attacks may be commonly used in other normal contexts.

A similar neural network with no hidden units also provided best attack classification performance. Classification performance on training data was not perfect because two stealthy versions of the fdformat attack, which contained no indications of the attack type, were misclassified as eject attacks. File and attack script names in these two stealthy attacks had been selected at random by the attackers and all attack exploit scripts and attack actions were hidden.

4. Results

Figure 2 shows performance across all 35 test attacks for three systems. One system is the baseline reference system using the total keyword counts of all old keywords. The second is an improved baseline system using additional generic keywords but simply counting keyword occurrences. The third is the same as the second system, but using discriminant neural network training to weight keyword counts. Performance is shown by plotting receiver operating characteristic curves or ROC's which show the detection rate versus the number of false alarms produced by each system per day. These curves are generated by varying a detection threshold for each system which determines the level a system's output score must exceed for a session to be labeled an attack. The detection rate is the number of attack sessions with scores above threshold divided by the number of attacks (35) and changed to a percentage, and the false-alarm rate is the total number of normal sessions above the threshold divided by the number of days in the training data (10). The detection threshold is swept over a range that produces a detection rate ranging from zero to 100% to generate each ROC.

Figure 2 shows that the baseline keyword counting system with old keywords requires a high false-alarm rate (greater than 50 false alarms per day) to detect more than 80% of the attacks. Adding new keywords lowers that false-alarm rate of the baseline keyword counting system to roughly 10 false alarms per day to detect 80% of the attacks and using a neural network to weight keyword counts of a smaller set of 30 keywords lowers the false- alarm rate to an acceptable and practical rate of roughly one false alarm per day. These results demonstrate that a dramatic reduction in false-alarm rates can be produced by a combination of adding new keywords and discriminative training. The availability of training data with ground-truth and both normal and attack sessions was essential to permit both selection of new keywords and discriminative training. These results also demonstrate that even well designed

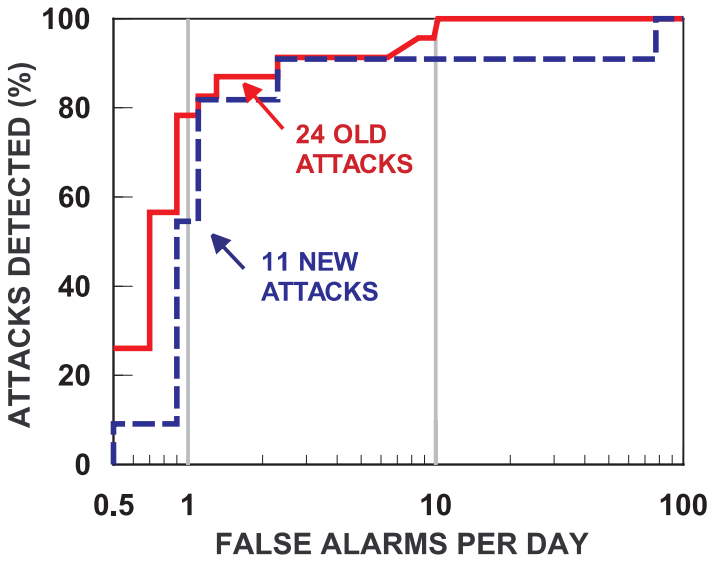

Figure 3.ROC's for old attacks from the training data and new attacks that occur only in test data.

keyword based systems can perform poorly if keywords are outdated and primarily specific to individual attacks.

Attack classification performance during testing was measured for the five attack types that were in the training data. Classification was perfect for attacks that could be identified solely by the identity of the victim operating system (loadmodule and perl) and when attack exploit scripts were clearly visible in telnet transcripts. All stealthy instances of ffbconfig and fdformat attacks were, however, misclassified as eject attacks.

Figure 3 shows the performance of the enhanced keyword system with new keywords and discriminant training for old attacks that occurred in the training data and for new, previously unseen, attacks that occurred only in the test data. Low false-alarm rates of roughly one false alarm per day are sufficient to detect 80% of both types of attacks. This surprising result demonstrates that the keyword-based intrusion detection system can generalize and

detect new attacks even when trained only on old attacks. Generalization occurs because this system is designed to detect attack setup actions, actions taken to hide attack exploit script transmission, and actions taken after a successful attack that are common to many attacks. Although the mechanism used to obtain root-level privilege changed between old and new attacks, these characteristics of the attacks were similar for old and new attacks.

Figure 4 shows performance of the improved systems with discriminative training and new keywords for attacks that were made stealthy by distributing them across multiple sessions and for normal attacks that were completed in a single telnet session. As can be seen, it is more difficult to detect attacks that are distributed across multiple sessions because each individual session contains fewer keywords. A false-alarm rate of roughly one false alarm per day is sufficient to detect 80% of the attacks that occur completely within one session. This rate must increase to roughly 10 per day to detect 80% of the multi-session attacks.

Figure 5 shows performance of the improved intrusion detection system for attacks that were made stealthy by hiding the contents of attack exploit perl, shell, and C source code scripts and for attacks where exploit scripts were clearly visible in telnet sessions. Although it is more difficult to detect attacks when attack scripts are hidden, 80% of both types of attacks can be detected at a false-alarm rate of roughly one false alarm per day. These results again show the benefit of including keywords that detect attack setup and post-attack actions instead of relying on attack-specific keywords visible primarily in attack scripts.

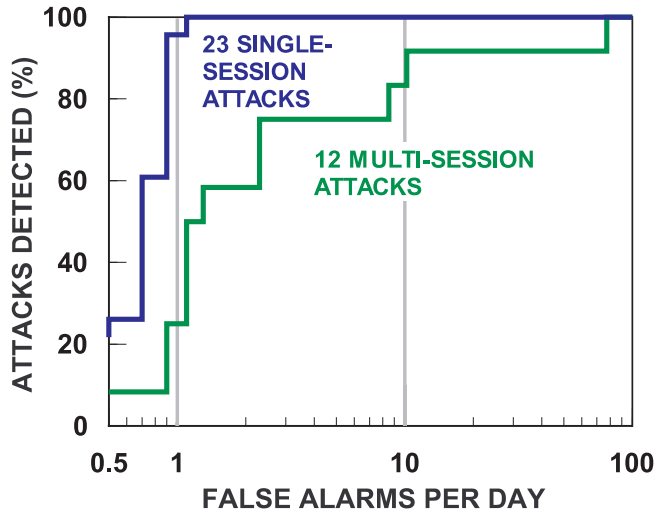

Figure 4.ROC's for normal attacks that were completed in one telnet session and stealthier attacks that were distributed across one or more sessions.

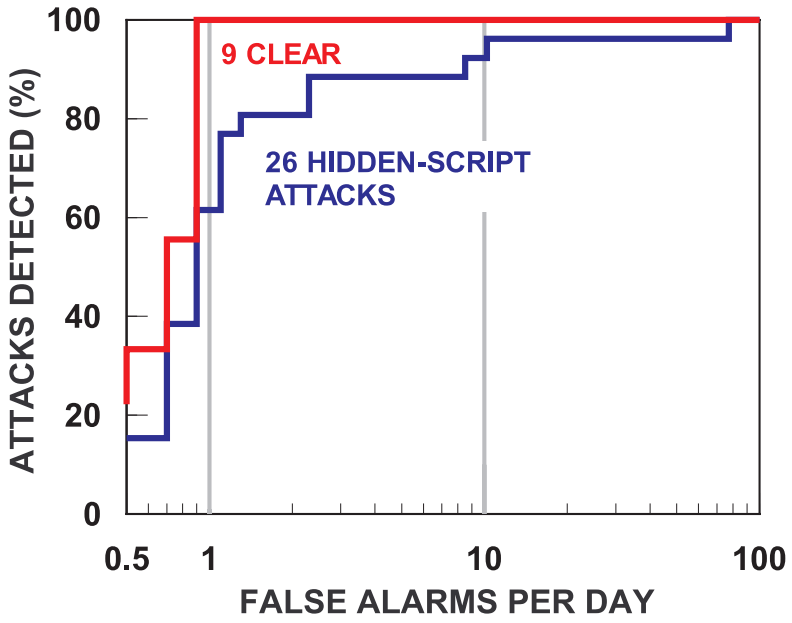

Figure 5.ROC's for attacks where the attack exploit script is clearly visible in the telnet session and for stealthier attacks where attack scripts are hidden.

5. Summary and Discussion

Large improvements in performance were obtained for a simple keyword intrusion detection system using a combination of adding new keywords and discriminative training. New keywords were added which detected actions that were common to many attacks and simple neural network discriminative training was used to produce output posterior probabilities which distinguish between telnet sessions with normal actions and with attacks. An improved system had a high detection rate of roughly 80% at a low false alarm rate of roughly one false alarm per day. Improved performance required using training data from the DARPA intrusion detection evaluation data base for both normal sessions and attack sessions. Attack sessions are necessary to derive new keywords and both normal and attack sessions are required to support discriminative training.

The improved system that uses new keywords and discriminative training could detect old as well as new attacks not included in the training data, and it could detect stealthy attacks where attack exploit scripts were hidden, and (to a lesser extent) attacks distributed across multiple sessions. Many existing keyword-based intrusion detection systems could be improved by adding similarly selected keywords and by using discriminative training to weight keyword counts. These modifications do not have to increase computation rates because the number of keywords does not necessarily need to be expanded and the keyword score weighting requires very little computation. Future work is planned to evaluate this approach for actual network traffic, to extend keywords to detect additional pre-and post- break-in actions, and to extend the system to integrate information across multiple telnet sessions and network services

Acknowledgements

This work was sponsored by the Air Force under Air Force Contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

References

1. Bishop, M., S. Cheung, C. Wee, J. Frank, J. Hoagland, and S. Samorodin (1997), “The Threat from the Net,” IEEE Spectrum, 34(8):56-63.

2. Heberlein, T. (1995). “Network Security Monitor (NSM) - Final Report”, U.C. Davis: February 1995, http://seclab.cs.ucdavis.edu/papers/NSM-final.pdf.

3. Cisco Systems, Inc. (1998). “NetRanger Intrusion Detection System Technical Overview,” http://www.cisco.com/warp/public/778/security/netranger/ntran_tc.htm.

4. Lawrence Livermore National Laboratory (1998). “Network Intrusion Detector (NID) Overview,” Computer Security Technology Center, http://ciac.llnl.gov/cstc/nid/intro.html.

5. Lee, W., S.J. Stolfo, and K. Mok (1999) “Mining in a Data-flow Environment: Experience in Network Intrusion Detection”, Proceedings 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99), San Diego, CA.

6. Neumann, P. and P. Porras (1999) “Experience with EMERALD to DATE”, in Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA.

7. Cunningham, R. K., R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman, (1999). “Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation,” in Proceedings ID'99, Third Conference and Workshop on Intrusion Detection and Response, San Diego, CA: SANS Institute.

8. Lippmann, R. P., D. Fried, I. Graf, et al. (2000) “Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation”, in press Proceedings of the DARPA Information Survivability Conference & Exposition (DISCEX'00).

9. Lincoln Laboratory, MIT (1999) “DARPA Intrusion Detection Evaluation”, http:// www.ll.mit.edu/IST/ideval/index.html.

10. Lippmann, Richard P., Linda Kukolich, and Eliot Singer (1993) “LNKnet: Neural Network, Machine Learning, and Statistical Software for Pattern Classification,” Lincoln Laboratory Journal, 6(2), 249-268.