A Methodology for Designing Accurate Anomaly Detection Systems

Kenneth L. Ingham
University of New Mexico
Department of Computer Science
Albuquerque, NM, USA 87131
ingham@i-pi.com

Anil Somayaji
Carleton University
School of Computer Science
Ottawa, ON, Canada K1S 5B6
soma@scs.carleton.ca

ABSTRACT

Anomaly detection systems have the potential to detect zero-day attacks. However, these systems can suffer from high rates of false positives and can be evaded through through mimicry attacks. The key to addressing both problems is careful control of model generalization. An anomaly detection system that undergeneralizes generates too many false positives, while one that overgeneralizes misses attacks. In this paper, we present a methodology for creating anomaly detection systems that make appropriate trade-offs regarding model precision and generalization. Specifically, we propose that systems be created by taking an appropriate, undergeneralizing data modeling method and extending it using data pre-processing generalization heuristics. To show the utility of our methodology, we show how it has been applied to the problem of detecting malicious web requests.

Categories and Subject Descriptors

D.2 [**SOFTWARE ENGINEERING**]: Security

General Terms

Security

Keywords

HTTP, Web Server Security, Anomaly Detection, Intrusion Detection

1. INTRODUCTION

Unlike the signature-based intrusion detection systems (IDSs) in common use, anomaly-based IDSs have the potential to detect previously unseen, or zero-day, attacks. However, anomaly detection systems can be evaded through carefully crafted attacks and they often produce a large number of false positives. To build successful anomaly detection systems, we must develop detection methods that are better at detecting attacks but without misclassifying legitimate behavior.

The key to this trade-off lies in the nature of the generalization performed by a given anomaly detection method. If the method generalizes too much over training examples, then it will be easy for attackers to craft attacks that resemble normal behavior; if it generalizes too little, then previously unseen but legitimate behavior will generate false alarms. Careful control of generalization, then, is central to solving both problems.

In this paper we present a methodology for developing anomaly detection methods that achieve both a low rate of false positives and a high attack accuracy. The key insight behind this methodology is that while it is easy to increase the level of generalization of a given method by filtering inputs, it is more difficult to reduce the the level of generalization. Thus, by starting with an undergeneralizing method and selectively adding input generalization filters, we can construct both lightweight and highly accurate anomaly detection mechanisms.

The rest of this paper is organized as follows. In Section 2, we discuss the generalization problem and examine how it has been addressed in past work. In Section 3, we describe in detail our design methodology for accurate anomaly detection systems. Section 4 describes how we applied this methodology to the development of web request (HTTP request) anomaly IDSs using both a finite automata-based and a n-gram based models, and Section 5 presents the performance of these systems. Section 6 discusses limitations and future work, and Section 7 presents additional related work. Section 8 concludes.

2. BACKGROUND

If an anomaly detection system is to do more than simply memorize its training data, it must generate a model that represents a set of examples; i.e., it generalizes. When an anomaly detection system generalizes, it accepts input similar to, but not necessarily identical to, instances from the training data set. In other words, the set of instances considered normal (the normal set) is larger than the set of instances in the training data. For most anomaly detection systems (for example, those used with web servers), the set of possible legal input is infinite and the complete normal set is not known and might be changing over time. In this case, the anomaly detection system must use an incomplete set of training data, and generalization is a requirement.

The goal for an anomaly detection system is a model that accurately describes normal behavior, as illustrated in Figure 1 (a). If the algorithm generalizes too much (**overgeneralizes**), then the normal set is too large. In this case, attacks "close" to the training data might be identified as normal (a false negative), limiting the usefulness of the system. Figure 1 (b) illustrates an overgeneralizing anomaly detection system. On the other hand, a system that simply memorizes the training data will need sufficient storage for the complete set of normal. Obviously, this is impossible when the normal set is unknown or infinite. When trained on a finite set, a

LANC'07 10-11 October 2007, San José, Costa Rica

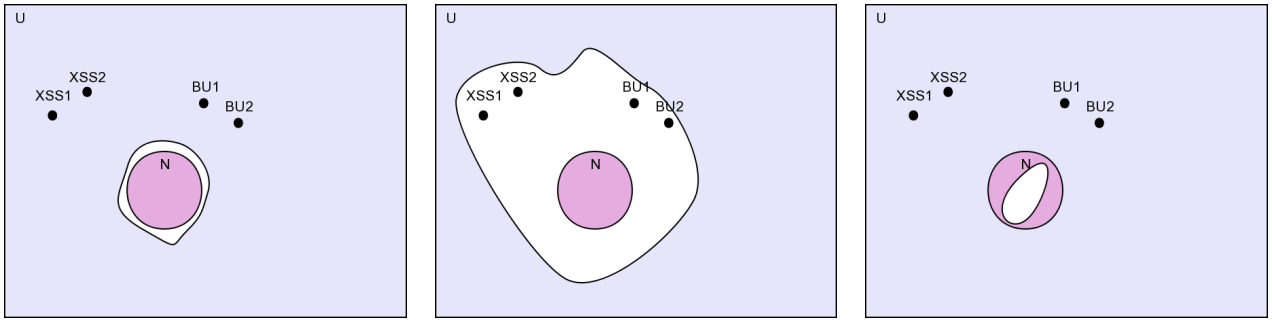


Figure 1: Diagrams illustrating (a) desired, (b) over-, and (c) undergeneralization. The points BU1, BU2, XSS1, and XSS2 represent attacks, N represents the set of normal inputs, and the line surrounds the set accepted by the anomaly detection system. U is the set of all possible inputs to the system.

system such as this would **undergeneralize**, and erroneously flag normal events as anomalous (false positives). An undergeneralizing system miscategorizes normal instances that are slight variants of training data. Figure 1 (c) illustrates undergeneralizing.

Undergeneralization manifests itself as the anomaly detection system memorizing distinct values that should be represented by a more general representation. For example, instead of memorizing all of the possible dates and times that appear in the header of a web request, better accuracy can be achieved by recognizing if the date and time follow the standard and indicating simply whether or not the time was well-formed or not.

Minimally generalizing, simple methods such as overlapping n-grams are well suited to modeling relatively predictable inputs such as the system calls produced by running processes [4]. More complex and variable behaviors, such as those observed at the network layer, have generally motivated researchers to look to methods that overgeneralize, however. For example, PAYL [11], a system for detecting anomalous network traffic, classifies packets using statistical distance measurements on two highly generalizing features: character distributions (one or two bytes) and packet lengths. Recently it was shown that attackers can evade PAYL by constructing packets that have similar character distributions and lengths to normal traffic [3]. Note that this attack is possible because PAYL's model permits packets that can be substantially different from normal behavior.

In order to achieve better attack coverage, anomaly IDSs have been developed that combine multiple, highly generalizing detectors in order to limit what kinds of behavior are acceptable [8]; in recent work, however, Ingham and Inoue [6] showed that such a strategy does not necessarily achieve a high attack detection rate if false positives are also minimized.

3. DESIGN METHODOLOGY

Rather than start with overgeneralizing measures and combine them to constrain acceptable behavior, we propose to instead start with undergeneralizing measures and then pre-process input features in order to achieve an appropriate balance between attack coverage and false positives.

Specifically, our design methodology consists of the following steps:

1. Choose an undergeneralizing modeling method—one that is capable of memorizing or near-memorizing individual input instances. For example, use an automata induction method.
2. Look for evidence of memorization of portions of specific input instances due to high input variability. In the case of a DFA, such instances can be identified as nodes with a large number of outbound edges, each with a low usage.
3. Use heuristic-based input filters in order to replace highly variable portions of the input with more constrained inputs.
4. Re-train the model and verify that the targeted model portions no longer encode specific inputs. If necessary, identify other inputs to be filtered and repeat.

While portions of this methodology can be automated, the choice and design of input filters is a highly complex task. We discuss the issue of automation further in Section 6.

4. MODELING WEB REQUESTS

To illustrate that our methodology can work in practice, we now present the design and evaluation of a web request anomaly-based IDS. As most attacks on web servers can be encoded in a single request, we chose to model individual HTTP requests in isolation. For an undergeneralizing modeling method, we chose to study two methods: one based upon n-grams of tokens for HTTP requests, and another based upon inducing a token-generating deterministic finite automaton (DFA).

Our n-gram models [2] were generated by sliding a window of length n across a string of tokens and storing the contents of each window. The result is a set of strings of length n. For example, given the string `abcdef` and $n = 3$, the resulting 3-grams are: `abc`, `bcd`, `cde`, and `def`.

To determine whether a given request was anomalous, we used the following similarity measure for the n-gram model:

$$s = \frac{\text{\# of } n\text{-grams from the request also in the training data}}{\text{\# of } n\text{-grams in the HTTP request}} \in [0, 1]$$

Following past practice and the results of our testing within this domain [5], we used $n = 6$ in our evaluation.

To create a DFA that classified tokenized HTTP requests, we used the Burge DFA induction algorithm [7], a one-pass, $O(nm)$ algorithm, where n is the number of samples in the training data set and m is the average number of tokens per sample. The algorithm does not require negative examples. To determine whether a given request was anomalous, we used this similarity measure for our DFA models:

$$\frac{\text{\# of tokens reached by valid transitions}}{\text{\# of tokens in the HTTP request}} \in [0, 1]$$

4.1 Identifying undergeneralization

As explained in Section 5, we observed that both methods produced very large models that had high false positive rates when they were trained on raw HTTP request tokens. As these were both evidence of undergeneralization, we the studied the learned models to identify tokens that were highly variable and so were being memorized.

Both the n-gram and DFA methods represent a HTTP request as a directed graph. In the digraph model that they use, the pattern indicating simple memorization is a node with a high out degree and low usage counts on all of the outbound edges. A good metric for identifying such structures is $\frac{o}{m}$ where o is the out degree of the node and m is the highest usage count (generated during DFA construction) of any outbound edge. Sorting nodes by this value highlights inputs where the anomaly detection system is attempting to memorize high-variability data. These are the portions of the model where additional generalization is needed.

4.2 Generalization Filters

Applying the techniques from Section 4.1, we found the following portions of a HTTP request where the models were effectively memorizing input tokens:

1. IP addresses, from lines added by proxies such as `Via` and `X-Forwarded-for` as well as occasionally in email addresses and URIs in the `Host` and `Referer` lines.
2. The `Referer` header field; this is a URI that led the user to the web site, and frequently is the result of a search. The search URLs are complex and highly variable.
3. Integer ranges (used for saving bandwidth by only sending the portions of the resource changed from a cached version). The values are integers and they range from 0 through the size of the largest file on the server, which in our training data is 22,788,278 bytes. These do not always fall on common block boundaries, as we see requests for a wide variety of values.
4. User agent additional information (e.g. user agent version, capabilities, etc).
5. Hashes (Entity tags, session IDs, etc). Note these are intended to be unique.
6. Dates, which appear in lines such as `If-Modified-Since`, `Unless-Modified-Since` and other related lines, as well as the `Date` header.
7. Host names, which appear in many of the same locations as IP addresses.

For each of these, we implemented generalization filters by modifying our HTTP parser to recognize and return whether the value was valid or not instead of the actual value. In the case of hashes, we only handled the most common (in our data) cases of PHP Session IDs and Entity tags.

5. RESULTS

To evaluate the utility of our methodology, we now present results that illustrate how our modeling methods work both with and without our empirically-derived, targeted generalizations. For more detailed information on how these and other related experiments were conducted, please refer to [5].

To test our anomaly detection system, we need datasets of both normal and malicious HTTP requests. In the results presented here, the normal requests were gathered from the University of New Mexico Computer Science (UNM CS) departmental web server. Background attacks in this data set were filtered out using a combination of *snort* rules and manual inspection. Our attack database contains 63 attacks, some of which are different examples of the same vulnerability—either a different exploit for the same vulnerability or an exploit for the vulnerability on a different operating system. The attacks came from a variety of sources and represent many classes of attacks against web servers or web applications, including buffer overflows, non-buffer overflow input validation errors, decoding errors, etc.

Figure 2 (a) shows that without the targeted generalization filters, the DFA and 6-grams are both unusable due to their high false positive rates—i.e., they undergeneralize. In order to achieve a true positive rate of only 80%, an administrator would have to accept a false positive rate of over 99% for either system. The results of using these generalizations together are in Figure 2 (b). Adding the identified generalizations results in a substantial improvement in accuracy; for the same 80% true positive rate, the false positive rate is 0.2% for 6-grams and 1.6% for the DFA.

6. DISCUSSION

When a system over- or undergeneralizes too much, the potential of an algorithm might be easy to miss, as shown by the difference between the DFA or 6-gram algorithms without and with the targeted generalizations applied. These generalizations make the difference between an algorithm with a too-high false positive rate and one that might be usable in a production system.

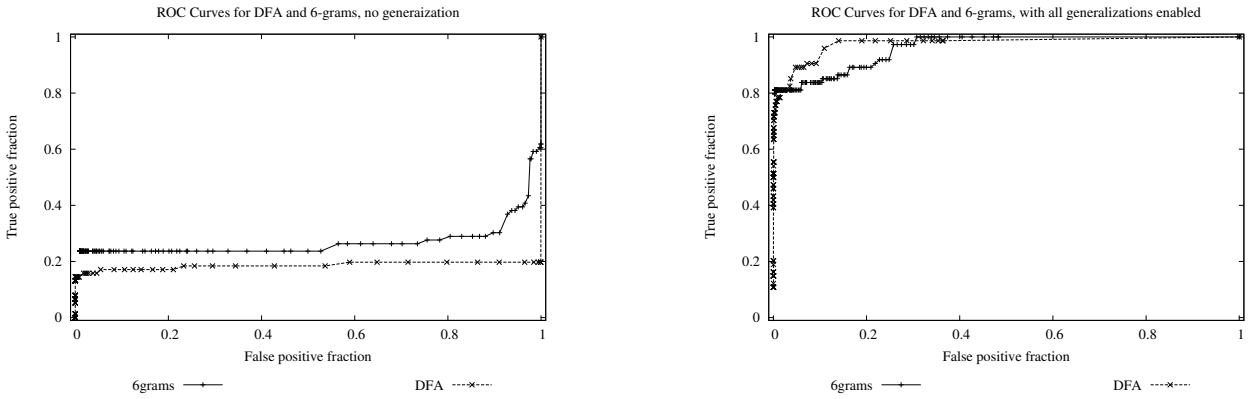
While significant generalization is necessary for accurate anomaly detection with data streams such as HTTP, it is not sufficient. The representation of the data stream must be appropriate for detecting interesting anomalies. For example, representing HTTP requests as the bits in the ASCII character string containing the request would likely result in poor accuracy.

The basic strategy of incrementally improving an undergeneralizing but precise model of behavior with automatically discovered, targeted generalizations is applicable to other domains where an anomaly detection system is modeling structured but highly variable, nonstationary data. Examples of such data streams are network protocols, application programming interfaces, and even system call arguments. We believe the application of the this design methodology to other program-level IDSs may produce systems that better walk the line between attack coverage and false alarms.

Our targeted generalizations were manually generated after the process identified the generalizations that would produce the largest benefit. With models such as ours that can be represented as graphs with tokens on transitions, identification of undergeneralizing areas is relatively straightforward and can be automated: just sort by degree of fan-out. Creating appropriate generalization filters, however, is a much more subtle problem. The key challenge is not how to generalize, but how to do so without permitting attacks. Verifying strict protocol adherence and basic semantics (e.g., valid date formats) could be automated if a precise protocol description is available. Such verification, however, may permit malicious input that is syntactically correct. Alternative strategies for targeted generalization that leverage contextual information is a potentially promising area for future work.

7. RELATED WORK

Overall, many researchers have mentioned generalization, but few have attempted to control it, understand it, or detail its relationship with accuracy. Li et al. [9] are some of the only researchers to investigate the relationship between generalization and anomaly detector accuracy. Noting that generalization can (at times) improve accuracy, and they showed that generalization on normal data is good, while generalizing on attack data is not. This result contra-



(a) (b)

Figure 2: Receiver Operating Characteristic curves showing the accuracy of the n-grams and DFA without (a) and with (b) the targeted generalizations.

dicts the work reported by Anchor et al. [1], who discussed generalizing detectors for attacks, with the goal of improving an anomaly detector at detecting new, similar attacks in network packet data. Robertson et al. [10] also generalized anomalies with the goal of generating "anomaly signatures" to find further attacks of a similar nature. None of these researchers characterized the amount of generalization, nor did they consider controlling generalization to improve accuracy.

8. CONCLUSIONS

Anomaly detection has the potential to detect novel attacks, but to succeed generalization must be controlled for accuracy. Undergeneralizing systems have problems with false positives, while overgeneralizing systems suffer from false negatives; systems that both over- and undergeneralize have problems with both.

By searching the model for indications of undergeneralizing and targeting generalizations to those specific locations in the structure of the input data, it is possible to substantially improve accuracy. This increased accuracy comes without the risk of overgeneralizing (hence reducing accuracy), assuming the heuristics for the targeted generalizations are carefully developed. Previously, researchers relied on their intuition about how to tweak an algorithm for good accuracy. Or, they attempted to select a large set of observables and try to get around the over- and undergeneralizing with statistical methods for combining the measures. By following the ideas presented in this paper, researchers can more quickly arrive at an accurate anomaly detection algorithm.

9. ACKNOWLEDGMENTS

Partial funding of K.I.'s research was provided by the National Science Foundation grant ANIR-9986555. A.S. wishes to also acknowledge the support of Canada's NSERC Discovery program and MITACS.

10. REFERENCES

[1] K. P. Anchor, J. B. Zydallis, G. H. Gunsch, and G. B. Lamont. Extending the computer defense immune system: Network intrusion detection with a multiobjective evolutionary programming approach. In *Proceedings of ICARIS 2002: 1st International Conference on Artificial Immune Systems Conference*, 2002.

[2] M. Damashek. Gauging similarity with n-grams: language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

[3] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. Polymorphic blending attacks. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 17–17, Berkeley, CA, USA, 2006. USENIX Association.

[4] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A sense of self for Unix processes. In *1996 IEEE Symposium on Security and Privacy, 6-8 May 1996, Oakland, CA, USA*, pages 120–128, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[5] K. L. Ingham. *Anomaly Detection for HTTP Intrusion Detection: Algorithm Comparisons and the Effect of Generalization on Accuracy*. PhD thesis, Department of Computer Science, University of New Mexico, Albuquerque, NM, 87131, 2007.

[6] K. L. Ingham and H. Inoue. Comparing anomaly detection techniques for HTTP. In *Recent Advances in Intrusion Detection*, 2007.

[7] K. L. Ingham, A. Somayaji, J. Burge, and S. Forrest. Learning DFA representations of HTTP for protecting web applications. *Computer Networks*, 51(5):1239–1255, 11 April 2007.

[8] C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5):717–738, 2005.

[9] Z. Li, A. Das, and J. Zhou. Model generalization and its implications on intrusion detection. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 222–237, 2005.

[10] W. Robertson, G. Vigna, C. Kruegel, and R. A. Kemmerer. Using generalization and characterization techniques in the anomaly-based detection of web attacks. In *Network and Distributed System Security Symposium Conference*

Proceedings: 2006. Internet Society, 2006.

[11] K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings*, volume 3224 of *Lecture Notes in Computer Science*, pages 203–222. Springer, 2004.