The Use of Information Retrieval Techniques for Intrusion Detection

Ross Anderson, Abida Khattak

University of Cambridge, Computer Laboratory
Pembroke Street, Cambridge CB2 3QG, UK
Email rja14,ak247@cl.cam.ac.uk

10th June 1997

Abstract. Intrusion detection is a broad problem, and we need a greater range of tools than is currently available. In this article, we report a new approach. We have applied information retrieval techniques to index audit trails. These indexes can be extremely efficient at detecting attacks whose signature is an unusual combination of events, and they may consume only a very small additional amount of storage. This approach allows the intrusion detection community to adopt a wide range of techniques developed in applications ranging from library science to web search engines.

1 Introduction

The computer security literature has up till now generally considered the intrusion detection problem in terms of network based attacks on distributed UNIX systems [5]. Researchers have recently started to realise that essentially similar problems are being tackled by the operators of mobile phones and credit card systems [21]. There are many other applications: detecting suspicious trades in capital and commodity markets, looking for symptoms of money laundering, scrutinising pharmacy records for indications of narcotics abuse, and even processing the last few years' accounts of companies applying for business loans. It is surely significant that system developers in many different application areas are tackling similar problems, yet appear unaware of each others' work.

Given this diversity, we should not be surprised to find that none of the known intrusion detection methods is sufficient alone; each addresses a different set of threats. But there are some general themes, and some techniques which work across a surprisingly wide range of applications.

Many intrusion detection problems can be characterised as 'finding a needle in a haystack': there is an enormous amount of data available, but resource constraints mean that only a tiny fraction of it can be examined. According to a report co-authored by a deputy director of the NSA, the biggest problem facing the signals intelligence community is traffic selection rather than the subsequent processing [18]. On a more mundane level, we visited a local store to examine its security measures: dozens of video camera feeds were watched by only two staff, who used 'gut instinct' to decide which shoppers to monitor.

In many applications, there are quantitative indicators of a successful intrusion. A casino manager reported that he only started scrutinising the video footage of his croupiers when the 'drop' fell below the expected level; he was unconcerned about any fraud that was too small to show up in the bottom line.

A more rigorous approach is taken in mainstream commerce. Perhaps the oldest established intrusion detection technique is double entry bookkeeping, which enables banks and other commercial enterprises to balance all their books at regular intervals and, in theory at least, detect any missing assets. This has been formalised in the Clark-Wilson security policy model [4] and is generally effective; the 1% or so of a bank's staff who try to embezzle money from it each year are mostly caught quickly, while the amount involved is still small [1].

The basic idea is not limited to commercial applications. There is often some invariant that will be disturbed by a successful intrusion, especially where our main concern is to protect the integrity, authenticity or availability of data. However, invariant-based techniques may be inadequate either because there is no invariant (a successful and covert breach of confidentiality that alters no data being an example), or where there is no complaint (as with so-called 'victimless crimes'), or when complaints are not made quickly enough (e.g., complaints from telephone subscribers usually arrive too late for phone phreaks to be caught). In such cases, we will need other methods.

Another very general intrusion detection technique is threshold detection. Concrete examples include raising an alarm after three false password attempts, when the number of international calls made by a subscriber exceeds ten in one day or when the call value exceeds $600, or on observing a sudden increase in narcotics prescriptions issued by a medical practice. This simple technique is often combined with application-specific methods. For example, stock exchange monitoring systems look for a surge in the volume of trades that occurs before a price move or price sensitive announcement; a surge after such an event is of little interest. In the classical intrusion detection literature, threshold detection is described as a component of the NSA's Multics Intrusion Detection and Alerting System (MIDAS) [25] and Los Alamos' Network Anomaly Detection and Intrusion Reporter (NADIR) [11].

Many researchers seek new general methods for intrusion detection, and their contributions are based on a wide range of techniques: state transition analysis [12, 13, 22], AI/expert systems [3, 11, 17, 26], statistical profiling [14], immune system models [7, 8], data mining techniques [19], and various mixtures of neural networks, genetic algorithms and fuzzy logic [10, 24]. Many of these approaches have not been robust, or at least it has been difficult to test their robustness [23]. A particular problem arises when an attacker masks his behaviour by performing large numbers of innocent actions between the malicious ones.

One of the motivating factors in developing such systems is the desire that users should not have to write code. Many fielded systems use hard code for establishing profiles for users or processes, as in anomaly detection, or for implementing expert rules in misuse detection. This code has to be changed when new attacks are reported. In some environments, it is assumed that users will write their own `perl` scripts to search audit trails. Such assumptions are usually false; the outcome is that the audit trails do not get examined at all.

Some systems aim to have quite general automatic methods to detect intrusions, even of previously unknown types, and to find these subsequently in all observed data. We feel that this is over-ambitious; we believe that the successful intrusion detection systems of the future will usually incorporate a number of techniques rather than a 'one-strategy-fits-all' approach, and the greater the variety of tools that are available, the better.

We also feel that the mechanisms whereby a new kind of intrusion is discovered, and those which enable instances of it to be detected subsequently, will in general be different. A number of mobile phone operators keep records of calls made for 40 to 45 days for the millions of their subscribers using an Informix relational data base. From time to time, neural network techniques are used to look for anomalies in samples of the data; other intelligence sources are also used, such as billing complaints from subscribers. When anomalies are found that correspond to new methods of attack, rules for detecting future occurrences are coded in SQL and added to the system. Using the neural network directly on the production data would be too slow and would generate too many false alarms; screening of detected anomalies by a human familiar with the application domain appears to be still a critical step [6].

Such a two-phase approach to the intrusion detection problem is realistic in many applications. For example, in the general case of network intrusion detection, new types of attack are usually encountered by only a small number of system administrators, who in turn report them to CERTs which broadcast the news to the wider community [2]. A similar process is involved in detecting new computer viruses; samples are taken to laboratories and disassembled by experts, who then issue new search strings or other instructions through established distribution networks [16].

So the parameters of the research problem we set ourselves were as follows. We wanted one or more general techniques that would enable us to 'find needles in haystacks' in a number of applications (though not necessarily all); that would be robust against attackers inserting innocuous cover transactions into the audit stream; and that would enable users to install appropriate detection mechanisms for newly indentified attacks without having to write code.

2 A New Approach

The new approach we have developed to intrusion detection uses the methods of information retrieval (IR). Now familiar to all because of popular search engines like AltaVista, IR was one of the first applications of computers and developed out of librarians' need to index large quantities of text. IR systems typically make use of inverted files as indexes that support efficient searching for combinations of keywords, together with algorithms that can learn a user's preferences and help refine searches ('relevance feedback') using methods such as Bayesian inference. The standard textbook is [9], and a collection of influential historical papers in reprinted in [15].

Information retrieval techniques differ from fashionable 'data mining' approaches in that they are based on indexing rather than machine learning. Our approach, like AltaVista's, is based on the more traditional approach of indexing and inverted files; we assume that the task of identifying new attacks is carried out externally (although IR techniques may also be useful to analysts charged with this task). Our goal may be thought of, loosely, as constructing an 'AltaVista for audit trails'.

How would this work?

2.1 Concept of operations

When a system administrator learns from CERT (say) that a particular combination of commands can be used to obtain a root shell on a particular type of machine, then he has two concerns: firstly, to apply the patch that fixes the vulnerability, and secondly to find out if the attack has been used against any of his systems. At present, this second task might involve writing a number of perl scripts to search through logs and audit trails, so it will not get done.

Our vision is that he will simply cut and paste the details of the attack into his intrusion detection system, which will then consult an inverted file containing an index of the relevant audit trails and tell him whether (and if so where and when) an attack took place. If for some reason this search gives too large a number of false alarms, then he can progressively refine his search by eliminating the obvious false alarms until either he finds the attack, or convinces himself with reasonable confidence that none took place.

Such a facility would clearly be very desirable for a system administrator to have. But can it be built economically, or at all?

2.2 A prototype system

We have built a prototype system by combining the Unix `lastcomm` command and the `glimpse` public domain information retrieval engine.

The UNIX `lastcomm` file contains information about all previously executed commands that are recorded in the `/var/adm/pacct` file in reverse chronological order. A sample of its output is:

```
lastcomm    ajh      lat/620   1.13 secs Thu Feb 24 15:02
pg          ajh      lat/620   5.91 secs Thu Feb 24 15:01
ln          miller   ttyp1     1.14 secs Thu Feb 24 15:01
ls          rob      ttyp1     0.02 secs Thu Feb 24 15:01
pg          sjz      ttyp1     0.33 secs Thu Feb 24 15:00
sendmail  F root     __        0.01 secs Thu Feb 24 15:00
```

Glimpse (Global IMPlicit Search), a search engine developed by the University of Arizona, provides indexing and query schemes for file systems [20]. It allows flexible full text retrieval including Boolean queries, approximate matching (i.e. allowing misspelling) and even searching for regular expressions. In a sense, glimpse extends agrep to entire file systems, while preserving most of its functionality and simplicity.

We use a simple `perl` script to process the UNIX `lastcomm` file into a set of files, one for each userid. It is then indexed using `glimpse`. The indexing involves building an index of each user of the commands in the audit trail occurs in the set of files we have indexed.

In order to find out whether a particular intrusion has taken place on our system, we simply type the relevant commands into `glimpse` which retrieves the relevant information. Manual inspection can then determine whether it was an attack or just a coincidence.

Example: CERT advisory CA-93.17 of November 11, 1993 warns of an X windows vulnerability that appears in xterm [2]. Versions of this command that operate as setuid root can used to overwrite the password file, using a script such as following

```
mknod foo p
xterm -lf  foo
mv foo junk
ln -s /etc/passwd foo
```

The log created by an xterm can be symbolically linked to the password file because of a race condition between the access permission check on the logging and the logging itself.

With our prototype an administrator who receives such an advisory can find out at once whether his system has been attacked by using such intrusion pattern by simply typing a glimpse command

```
glimpse  -w 'mknod; xterm; ln'
```

Glimpse will now return any location where this combination of commands has been used.

3 Advantages of this approach

Firstly, if the problem is one of 'finding needles in haystacks' then IR techniques are the natural first choice, as they have developed over more than forty years for precisely this purpose.

Secondly, the attacker can gain little benefit by masking his attack with a large amount of innocuous padding traffic. The search query can be set to find matches at any desired distance. The distance will depend on the type of attack (with race conditions, as in the above example, it will be time limited but this will not always be the case). The cost of this flexibility depends on the implementation detail but is usually low. This is highly significant as robustness in the face of padding is one of the principal vulnerabilities of existing systems.

Thirdly, our approach is flexible in that it can search for a wide range of patterns, not all of which can be predicted in advance. Many of the techniques currently adopted by the intrusion detection systems rely on prior knowledge of the intrusion patterns. If a new intrusion pattern is detected then one has to write code in order to incorporate it in the system. As remarked above, users cannot be relied on to write code, and if it is written centrally then there may be significant delays in fielding it.

Information retrieval techniques also provide the flexibility to deal with inexact and multiple matches, such as where commands have variants (e.g. `foo`, `foo*`, `foo`). The existing features of `glimpse` can cope with approximate matching and with regular expressions, while the IR literature explores a number of other approaches.

So basing intrusion detection on IR techniques meets the three goals set out at the end of section 1. Of course, whether it is worthwhile in any given application will depend on performance aspects, ranging from its type 1 and type 2 error rates (precision and recall, in IR terminology) as well as the resource requirements.

A feature of `glimpse` that is of particular relevance for intrusion detection is that its index is compact (typically 2–4% of the indexed matter). A full inverted file would take as much space as the original; instead, `glimpse` divides its target file space into 256 sections and merely records in its index which of these sections a given word has appeared in. After looking up the index, it then reads the relevant sections and extracts the required data. The implication of this approach for intrusion detection is that full audit files can be kept on local machines, and a central index to them can be small enough to be manageable. This may often be enough to guard against attackers who penetrate a local system and then remove all trace of their attack from the log; an inconsistency between the index and the local file will provides the required indication of an intrusion.

The marginal cost of implementing a system along the lines described above may be zero in applications where audit formats need conversion anyway; an inverted file format is as good as any, and a compressed inverted format adds little overhead. The downside of this strategy is that if an attack involves a combination of very common commands, which appear in many sections of the audit space, then there would be a large number of false alarms. In such a circumstance, fuller indexing would be better. In any case, disk space is cheap.

We do not claim that our technique is a silver bullet that will provide better intrusion detection in all applications. On the contrary, many problems appear better served by traditional threshold methods: banks will continue to balance their books, and system administrators will continue to look at accounts on which a number of failed login attempts have been made. However, in the significant number of applications where the signature of an attack is the unusual combination of a number of observable events, methods based on our work may turn out to be unbeatable. They can also be combined with other methods; for example, a special token could be written to the index whenever several wrong passwords are entered, or whenever a bank branch records 'tellers' differences'. One can then search for significant combinations of such events.

We intend to develop a fuller scale implementation of IR-based intrusion detection, using the more complete data available (for example) from the Solaris `praudit` facility and `tcpdump`, and by looking at audit trails of access by hospital staff to patient records.

4 Conclusions

We have developed an innovative approach to intrusion detection, based on using information retrieval techniques to index audit trails. This enables a large body of established research to be brought to bear on difficult and important computer security problems.

Our preliminary work indicates that IR based intrusion detection works, and has the potential to be much more robust than the alternatives. It is the natural way to go about 'finding needles in haystacks', and where the signature of an attack is an unusual combination of observable events, it may well be unbeatable.

Acknowledgement: We are grateful to Karen Spärck-Jones, leader of the Computer Laboratory's Information Retrieval group, for introducing us to her subject and teaching us some of the tricks of her trade.

References

1. RJ Anderson, "Why Cryptosystems Fail" in *Communications of the ACM* vol 37 no 11 (November 1994) pp 32–40
2. "CERT Advisories," Available by Anonymous ftp from `cert.org/pub/cert_advisories`
3. K Chen, SC Lu, HS Teng, "Adaptive Real-Time Anamoly Tetection Using Inductively Generated Sequential Patterns," in *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pp 240–258
4. DD Clark, DR Wilson, "A Comparison of Commercial and Military Computer Security Policies", in *Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy*, pp 184–194
5. DE Denning, "An Intrusion-Detection Model," in *IEEE Transactions on Software Engineering*, v SE-13, no 2 (February 1987)
6. Proceedings of *Fighting Mobile Fraud*, 15–17/6/97, IBC UK
7. S Forrest, AS Perelson, L Allen, R Cherukuri, "Self-Nonself Discrimination in a Computer," in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pp 202–212

8. S Forrest, SA Hofmeyr, A Somayaji, TA Longstaff, “A Sense of Self for Unix Processes”, in *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pp 120–128
9. WB Frakes, R Baeza-Yates, *‘Information Retrieval Data Structures & Algorithms’*, Prentice-Hall, 1992.
10. S Goonatilake, “Intelligent Fraud Detection Systems — Benefits and techniques”, in [6]
11. J Hochberg, K Jackson, C Stallings, JF McClary, D DuBois, J Ford, “NADIR: An Automated System for Detecting Network Intrusion and Misuse,” *Computers & Security*, v 12 no 3 pp 235–248
12. K Ilgun, “USTAT: A Real-time Intrusion Detection System for UNIX,” *Proceedings of 1993 IEEE Symposium on Research in Security and Privacy* pp 16–28
13. K. Ilgun, A. Kemmerer, and P. Porras, “State Transition analysis: A Rule-Based Intrusion Detection Approach,” *IEEE Transaction on Software Engineering*, Vol. 21, No. 3, March 1995, pp. 181–199, IEEE, 1995.
14. HS Javitz, A Valdes, “The SRI Statistical Anomaly Detector,” in *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy* pp 316–326
15. K Spärck-Jones, P Willett (editors), *‘Readings in Information Retrieval’*, Morgan Kaufmann 1997
16. JO Kephardt, “A Biologically Inspired Immune System for Computers”, in *Artificial Life IV*, MIT Press 1994
17. M Levi, P Bissel, T Richardson, *‘The Prevention of Cheque and Credit Card Frauds’*, Crime Prevention Unit: Paper No. 26, London Home Office, June 1991.
18. S Landau, S Kent, C Brooks, S Charney, D Denning, W Diffie, A Lauck, D Miller, P Neumann, D Sobel, “Codes, Keys and Conflicts: Issues in US Crypto Policy”, *Report of the ACM US Public Policy Committee*, June 1994
19. W Lee, SJ Stolfo, “Data Mining Approaches for Intrusion Detection”, in *Usenix Security 98* pp 79–93
20. U Manber, S Wu, “Glimpse: A Tool to Search Through Entire File Systems,” TR 93-34, October 1993, Department of Computer Science, The University of Arizona, Tucson Arizona, available by ftp from from `glimpse.cs.arizona.edu`
21. *‘National Info-Sec Technical Baseline: Intrusion Detection and Response’*, Lawrence Livermore National Laboratory, Sandia National Laboratory, October 1996
22. PA Porras, RA Kemmerer, “Penetration State Transition Analysis: A Rule-Based Intrusion Detection Approach,” in *Proceedings of the Eighth Annual Computer Security Application Conference*, San Atonio, Texas, December 1992, pp 220–229
23. NJ Puketza, K Zhang, M Chung, B Mukerjee, “A Methodology for Testing Intrusion Detection Systems,” in *IEEE Transaction on Software Engineering* v 22 no 10 (Oct 1996) pp 719–729
24. N Ryman-Tubb, “Using Intelligent Software in Telecomms Fraud Detection and Management”, in [6]
25. MM Sebring, E Shellhouse, ME Hanna, RA Whitehurst, “Expert System in Intrusion Detection: A Case Study,” in *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988, pp. 74–81
26. HS Vaccaro, GE Liepins, “Detection of Anomalous Computer Session Activity,” in *Proceedings of the 1989 IEEE Symposium on Research in Security and Privacy* pp 280–289