Web Attack Detection Using ID3*

Víctor H. García, Raúl Monroy, and Maricela Quintana

Computer Science Department
Tecnológico de Monterrey, Campus Estado de México
Carretera al lago de Guadalupe, Km 3.5, Atizapán, 52926, Mexico
{A00471866,raulm,mquintana}@itesm.mx

Abstract. Decision tree learning algorithms have been successfully used in knowledge discovery. They use induction in order to provide an appropriate classification of objects in terms of their attributes, inferring decision tree rules. This paper reports on the use of ID3 to Web attack detection. Even though simple, ID3 is sufficient to put apart a number of Web attacks, including a large proportion of their variants. It also surpasses existing methods: it portrays a higher true-positive detection rate and a lower false-positive one. The ID3 output classification rules that are easy to read and so computer officers are more likely to grasp the root of an attack, as well as extending the capabilities of the classifier.

1 Introduction

In order to stay in business, many companies have a Web site, through which they promote or offer their products and let a candidate customer compute prices, compare product features, and so on. While profitable, these kinds of Web sites come along with a serious security problem, which is often approached via a standard protection schema: a firewall allowing HTTP(S) traffic. This schema is certainly not enough. According to S21SEC, 1320 public security vulnerabilities, out of the 2113 found only from June to November 2003, 62.5%, had their root in a Web application.

To better protect their resources, companies have strengthened their security mechanisms by incorporating clear and force-able security policies and by including other mechanisms such as an Intrusion Detection System (IDS). Intrusion is any action that puts on risk the integrity, confidentiality and availability of the company information. An IDS is a system that aims to detect intrusions on the fly while identifying the source of the attack.

Depending on its characterisation of intrusion, an IDS is of either of two types: i) misuse, and ii) anomaly. A Misuse IDS (MIDS) annotates as an attack any known pattern of abuse. MIDSs are very effective in detecting known

* We are grateful to the anonymous referees for their useful comments on an earlier draft of this paper. This research was supported by three grants: FRIDA, CONACYT 47557 and ITESM CCEM-0302-05.

attacks; they exhibit a high true positive detection rate. Yet, they are bad at detecting novel attacks. An Anomaly IDS (AIDS) annotates as an attack any activity that deviates from a profile of ordinary behaviour. Unlike MIDSs, AIDSs are capable of detecting novel attacks. However, they frequently tag ordinary behaviour as malicious, yielding a high false positive detection rate.

Depending on the activity it observes, an IDS can be placed at either of three points: a host, a network or an application. A host IDS usually audits the functionality of the underlying operating system, but can also be set to watch critical resources. An application IDS scrutinises the behaviour of an application. It commonly is designed to raise an alarm any one time the application executes a system call that does not belong to a pre-defined set of system calls, built by some means, an object-code analysis. An network IDS analyses network traffic in order to detect mischievous activities within a computer network. A denial of service attack resulting from flooding a network with packets can be pinpointed only at this level.

Current IDSs are easy to bypass: there is a number of means to get full access to a Web service. Tools have even been developed to go around IDSs, e.g. nikto, fragroute, ADMutate. The main problem of existing IDSs is that they cannot detect new kinds of attacks or even variations of existing ones (so-called *mimicry* attacks). This problem prevails in well known IDSs, like snort. To get around this problem, IDS researchers have turned their attention to machine learning techniques, including classification rules and neural networks.

Current IDSs are also easy to overrun, due to the staggering amount of information they must analyse. The root of the problem is that in a computer site there usually is one IDS: the omnipotent, global sensor. For example, it is a standard practise to have only one NIDS to analyse all the traffic generated over a company network, yielding an increase in the false positive detection rate. To get around this problem, IDS researchers have recently suggested one should have one sensor for each company site service, such as HTTPS, making an application IDS more specialised. Service-oriented IDS have the advantage of specialisation: they produce a low rate of both false positives and false negatives, but at the expense of having a number of small IDSs, possibly working without any coordination. So far, there are only a few publicly available service-oriented IDS [10].

In this paper, we introduce an IDS for protecting a Web application with a low missing alarm and false alarm rates. This IDS makes use of ID3, a well-known classifier that builds a decision tree from a fixed set of examples. Each input example is a Web application query; it has several attributes and belongs to a class, either attack or normal. As we shall see later on in this paper, ID3 requires only that the Web application queries be slightly pre-processed before application. Unlike a neural network, ID3's decision tree can be easily explored to find out the rules applied by the classifier. In our experiments, ID3 was able to successfully classify unseen Web application queries as an attack. If the data base training is growing up because of new vulnerabilities, ID3's performance does not change at all. Unlike neural networks, ID3's output is easy to read

by computer officers without having previous knowledge about decision trees techniques. Our hypothesis is that ID3 suffices to generalise on specifying a wide range of Web attacks, provided it is given a sufficiently large set of attack examples.

Paper Overview In what follows, we briefly describe related work, §2, and then characterise the kinds of attacks we want to detect, §3. Then, after outlining ID3 and the requirements it imposes on input data, §4, we show how to apply ID3 to build a decision tree for intrusion detection, §5. We recap and assess the results obtained throughout our experiments on validating our intrusion detection method, §6. Finally, we discuss the conclusions withdrawn from this research work.

2 Related Work

There are many IDSs currently available, ranging from commercial products to unprofitable ones. We briefly describe some of them below.

Rule Induction The application of inductive learning to intrusion detection has a long history in computer security. In 1990, e.g., Teng et al. [11] applied a time-based induction machine in order to characterise an audit trail as chunks of temporally co-related entries, yielding rule-based sequential patterns suitable for anomaly detection.

Cohen's RIPPER [4] is the crux for intrusion detection in JAM [10]. JAM is a distributed IDS that uses a collection of agents to form a number of intrusion detection models. These models are all merged into one, from which a meta-classifier is extracted. This meta-classifier is then used to detect intrusions; it can be combined with the meta-classifiers built in other sites and can migrate along them.

MADAM ID [5] is a framework that applies induction techniques (classifiers, association rules, frequent episodes, etc.) to build models for intrusion detection. It builds two kinds of models; one is about normal behaviour and the other about intrusions. Data is first pre-processed in order to find a representation where each event is normalised to a fixed number of attributes. Techniques are then used to find patterns of frequent occurrence, represented via association rules (connecting event features) and frequent episodes (connecting events).

Bayesian Classifiers Bayesian networks have proved to be very powerful to build decision models that operate under uncertainty conditions. When they are used to approach intrusion detection, an IDS amounts to a set of relations of conditional probabilities, as opposed to a set of classification rules. EMERALD [13], an IDS developed at SRI, includes a module, called eBayes TCP, that applies Bayesian networks to analyse traffic explosions.

ADAM [2] also uses a Bayesian network to build a profile of normal network activity. On operation, ADAM uses a sliding window to take observations from

the last D connections. These observations are then compared against the profile of normal behaviour, filtering-out items taken to be normal. The remaining data is then passed onto the misuse, naïve Bayes classifier. ADAM performs especially well with denial of service and probe attacks.

Amor et al. [1] compared naïve Bayes against decision tree algorithms at detecting intrusions at a host level, using the DARPA attack repository as a testbed. Both methods showed a similar detection rate (the latter technique being slightly better than the former one.) As expected, the construction of a naïve Bayes classifier is much faster than the construction of a decision tree one.

Support Vector Machines A Support Vector Machine (SVM) is a technique that has been used widely for both supervised and unsupervised learning. Mukkamala et al. [6] used 5 SVMs to approach intrusion detection. One SVM was used for separating normal traffic and the other ones for identifying each of the 4 attacks involved in the data test set of the 1999 KDD cup. Mukkamala et al. showed that SVMs surpass neural networks on this classification task.

Neural Networks More related to ours is Torres's work on Web intrusion detection [12]. His method, IDS-ANN, which uses an Ellman neural network, analyses data using a layered approach and, thus, it is very time-consuming. A neural network is a black box and so it is difficult to extract general knowledge about intrusion detection.

In this paper, we aim to test how well standard data mining techniques are up to characterise malicious Web application queries. We propose to use ID3, a decision tree technique, instead of a Bayesian network. This is because, according to Amor et al's results (see above), in this problem decision tree techniques perform slightly better at the expense of requiring more computational efforts. This extra cost is an issue when the objects to be classified are large, which is not our case (Web application queries are rather short sequences of symbols.) So we have chosen to favour ID3.

As we shall see later on in this paper, ID3 needs only that the Web application queries be pre-processed before being used. ID3 is able to correctly classify unseen Web application queries as an attack. If the data base training is growing up because of new vulnerabilities, ID3's performance does not change at all. Unlike neural networks, ID3's output is easy to read by computer officers without having previous knowledge about classification techniques. ID3 is widely available and is not difficult to implement.

In the next section we survey the flaws commonly exploited in a Web attack.

3 Web Attacks

SecurityFocus[2] analysed 3,000 security incidents related with Web servers, CGIs and other Web applications. They concluded that the exploits or other tech-

[2] `http://www.securityfocus.com`

niques used to perpetrate these attacks took often advantage of security vulnerabilities already published. This was either because the vulnerabilities were never fixed through a patch, or because the patch was not able to detect small variations of the exploits.

Thus, there are no radically new types of vulnerabilities that the exploits take advantage of. The exploits we want to detect are of four kinds: i) SQL Injection; Cross Site Scripting (XSS); iii) Code injection; and Directory Traversal.

3.1 SQL Injection

SQL injection is a kind of vulnerability where the attack tries to manipulate data base applications by issuing crafted SQL queries. The source usually is an incorrect escaping of dynamically-generated string-literals embedded in SQL statements.[3] The effect is that the SQL statement may do more than the application author intended.

For example, in a Web site, a user usually needs to type in his login and his password in order to get access to some application. With SQL injection, it is possible to send a crafted SQL query in order to bypass this kind of authentication:

```
http://localhost/login.cgi?id_user=vh';%20--
```

The fault exploited by this attack query has to do with the double hyphen (--). SQL ignores anything following a double hyphen, and thus the application, if not properly written, may take the user as valid, granting him full access.

3.2 Cross Site Scripting

Cross site scripting (XSS) is a vulnerability of Web applications that can be used by an attacker to compromise the *same origin policy* of client-side scripting languages, like JavaScript.[4] XSS occurs when a Web application unknowingly gathers malicious data on behalf of a user, usually in the form of a hyperlink. Usually the attacker will encode the malicious portion of the link to the site so the request looks less suspicious. After the data is collected by the Web application, it usually creates an output page for the user containing the malicious data that was originally sent to it, but in a manner to make it appear valid content from the website.[5]

Using XSS and social engineering, an attacker could steal information about credit cards numbers or other personal data. The following link is for public use when testing for XSS:

```
http://www.vuln-dev.net/<script>document.location='http://www.repository.
com/cgi-bin/cookie.cgi? '%20+document.cookie </script>
```

[3] http://en.wikipedia.org/wiki/SQL_injection
[4] http://en.wikipedia.org/wiki/Cross-site_scripting
[5] http://199.125.85.46/articles/xss-faq.shtml

3.3 Code Injection

Code injection is a kind of vulnerability of Web server applications that can be used by the attacker to make the Web application execute arbitrary code. It is a cracking technique used to obtain confidential information or get unauthorised access to a system. These are two example code injection queries:

1. `http://localhost/scripts..%c0%afwinnt/system32/cmd.exe?/c+dir`
2. `http://host/index.asp?something=..\..\..\..\WINNT\system32\cmd.exe?/c+DIR+e:\WINNT\*.txt`

3.4 Directory Traversal

Directory traversal is one of the most common attacks. It aims to traverse the directory structure of a Web server to access files that may not be public. Directory traversal exploits insufficient security validation of user supplied input file names, so that special characters used to traverse to a parent directory are passed through to the file APIs.[6] Two examples of directory traversal attacks are:

1. `http://host/cgi-bin/vuln.cgi?file=../../../../etc/motd`
2. http://host/cgi-bin/vuln.cgi?page=../../../../bin/ls%20-al%20/etc|

The second query, for example, requests for a full directory listing of the "`etc`" directory within a Unix system. It is possible to make different variations on these attacks in order to fool conventional IDSs.

These kinds of attacks are all Web application queries. They take the form of well defined strings. In what follows, we briefly describe the ID3 algorithm and, in the next section, we show how to apply it in order to recognise attack queries.

4 ID3

ID3 is a simple inductive, non-incremental, classification algorithm [7]. Using a top-down, greedy search through a fixed set of examples, it builds a decision tree, which is then applied for classifying future samples. Each example has several attributes and belongs to a class. Each non-leaf node of the decision tree is a decision node, while each leaf node corresponds to a class name.

ID3 extends the concept learning system algorithm adding a feature selection heuristic. Feature selection is used to identify the attribute that best separates the set of input examples, called the *training set*. If the selected attribute completely classifies the training set, then we are done. Otherwise, ID3 is recursively applied, in a greedy fashion, to identify the next best attribute.

When deciding which attribute is the best, ID3 uses a measure called information gain. Information gain is defined in terms of the amount of information portrayed by an attribute, called entropy in information theory. This attribute selection method is very powerful. ID3 is well-established in both industry and academia.

[6] `http://en.wikipedia.org/wiki/Directory_traversal`

ID3, however, operates only on examples described by the same attributes. Attributes must take values from a fixed, finite set. ID3 is not tolerant to noisy or missing attributes. Classes must be sharply defined.

4.1 Other Inductive Classification Algorithms

CN2 [3] is other inductive, classification algorithm, which outputs an (un)ordered list of classification rules, instead of a decision tree. When first proposed, CN2 also used information entropy as a feature selection heuristic. Later, the use of the Laplacian error estimate was suggested as an alternative evaluation function.

C4.5 is a decision tree generating algorithm [8]. It extends ID3 in two main respects: i) it handles training data with missing attribute values; and ii) it handles attributes that take values from an infinite, continuous range.

We ruled out the application of C4.5 to detecting attacks in Web application queries. This is both because Web application queries have no missing attribute values, and because attributes take values from a discrete range. CN2 is just as good as a choice for our problem. Indeed, one advantage of using CN2 over ID3 is that a computer officer would not have to explore the ID3 decision tree in order to obtain the rules applied by the intrusion detection method. In what follows, we show how to apply ID3 (or CN2) to our intrusion detection problem.

5 Generating a Decision Tree for Intrusion Detection

In this section, we describe how to build a decision tree for Web attack detection using ID3.

5.1 Normalisation of URL Locations

Given that they do not fulfil the ID3 data requirements, Web application queries need to be transformed, prior to the application of ID3. A Web application query involves a specific Uniform Resource Locator (URL) and a collection of reserved symbols. An URL is a string, conforming to a standardised format, that is used for referring to resources on the Internet, by their location.[7] The location is irrelevant from an attack formation perspective.

Thus, our first step is to transform every Web application query so that each of its attributes takes a value from a fixed, finite set. This is accomplished by parsing every input query so as to divide it into substrings, using "." and "/" as terminal symbols. Then, for each substring, if it does not match one string that we have previously marked as **reserved**, we replace it with the string "@". Otherwise, the substring is left unmodified. For example, the following Web requirement:

```
B?variableB=something&variableB2=../dir
```

is transformed to:

```
@?@=@&@=../@
```

Here, ?, &, = and .. are assumed to be reserved symbols.

[7] `http://en.wikipedia.org/wiki/URL`

5.2 Handling Queries of Different Size Using a Sliding Window

The resulting Web application queries cannot yet be input to ID3 since they are not specified by the same attributes. Queries, as illustrated in Section 3, are of different size. We get around this problem using a sliding window. The sliding window is slided one by one, thus this way we get examples with the same number of attributes. After experimenting with a window of different sizes (5, 8, 10, 12, 15), we chose to use a sliding window of size 10, since it proved to build a decision tree that more precisely captured a subset of our examples.

5.3 The Training Data Set

We gathered 400 Web application attack queries from three security vulnerability lists: i) Securityfocus, iii) Unicode IIS Bugtraq,[8] and iii) Daily's Dave vulnerability disclosure list.[9] We also gathered 462 Web application non-attack queries. Non-attack queries were gathered from the Apache log files of 3 servers.[10] Each query was then given one of five classes: i) SQL injection, ii) cross site scripting, iii) code injection, iv) directory transversal and v) normal.

We apply the sliding window strategy to the 862 example training set. Then, the resulting objects were input to ID3 and so we built a decision tree. The decision tree was made classify a number of not previously considered Web application queries. The results of this validation step are reported below.

6 Validation Stage: Experimental Results

Once built, the ID3 decision tree was used as an intrusion detection method, we call *ID3-ids*. We tested Id3-ids against a collection of real Web application queries. The attacks were output by two attack generation engines: i) nikto and ii) nessus. The attacks were input to ID3-ids via a Web proxy cache, Squid. Squid accepts any kind of Web query and then redirects it to a custom Web server. We compared ID3-ids with snort and IDS-ANN.

Tables 1—4 summarise the results obtained throughout experimentations. They indicate both the false alarm rate and the missing alarm rate, considering two sets of attacks. One set of attacks, generated by nikto, contains 1771 examples. The other, generated by nessus, contains 14594 examples. ID3-ids surpasses snort. IDS-ANN is slightly better than ID3-ids at distinguishing the kind of attack under consideration. However, the detection rate of ID3-ids (considering only attack or non-attack) is slightly better.

These results show that ID3 is a competitive alternative for detecting Web application attack queries.

---

[8] `http://www.securityfocus.com`

[9] `http://www.immunitysec.com`

[10] `http://www.zionn.org,http://www.ganexx.org,http://www.badc0d3d.org.ar/`

Table 1. ID3-ids vs snort performance on attacks generated by Nikto

Method name	1771 Nikto generated attacks			Detection rate
	detected	undetected		
		missing alarms	false alarms	
snort	1282	329	160	72.3%
ID3-ids	1650	77	44	93.2%

Table 2. ID3-ids vs snort performance on attacks generated by Nessus

Method name	14594 Nessus generated attacks			Detection rate
	detected	undetected		
		missing alarms	false alarms	
snort	10256	2789	1549	70.27%
ID3-ids	13668	686	240	93.65%

Table 3. ID3-ids vs IDS-ANN performance on attacks generated by Nikto

Method name	1771 Nikto generated attacks			Identification rate	Detection rate
	detected	undetected			
		missing alarms	false alarms		
IDS-ANN	1680	41	50	83.8%	94.86%
ID3-ids	1650	77	44	77.25%	93.2%

Table 4. ID3-ids vs IDS-ANN performance on attacks generated by Nessus

Method name	14594 Nessus generated attacks			Identification rate	Detection rate
	detected	undetected			
		missing alarms	false alarms		
IDS-ANN	13200	989	405	78.5%	90.44%
ID3-ids	13668	686	240	77.25%	93.65%

7 Conclusions and Further Work

ID3 is an effective means for detecting and classifying web application attack queries. It yields a 4.7% missing alarm (false positive detection) rate and a 1.6% false alarm (false negative detection) rate. One major drawback of ID3-ids is that it is non-incremental. So the decision tree has to be built on a regular basis, using an updated attack signature database. Unlike a neural network, the ID3 decision tree can be easily explored and so intrusion detection rules can be further refined by a computer officer. CN2 is as applicable as ID3 to this problem. Actually, we also conducted these experiments using the CN2 toolbox.[11] As expected, we obtained similar results using the information

[11] `http://www.cs.utexas.edu/users/pclark/software.html`

gain feature selection. The false negative and false positive detection rate, however, worsen when the Laplacian error estimate was selected.

Further work involves applying CN2 or ID3 to masquerader detection [9]. We plan on further validating ID3-ids with other attack generating frameworks, like canvas and core impact.

The attack database, the decision tree and the code developed within this research work are at available at `http://webdia.cem.itesm.mx/ac/raulm/pub/id3-ids/`.

References

1. Amor NB, Benferhat S, Elouedi Z (2004) Naive bayes vs decision trees in intrusion detection systems. In Omicini A, Wainwright RL (eds) *Proceedings of the 2004 ACM symposium on Applied computing*, pages 420–424. ACM Press
2. Barbará D, Couto J, Jajodia S, Wu N (2001) ADAM: A testbed for exploring the use of data mining in intrusion detection. *SIGMOD Record*, 30(4):15–24
3. Clark P, Niblett T (1989) The CN2 induction algorithm. *Machine Learning*, 3:261–283
4. Cohen WW (1995) Fast effective rule induction. In Prieditis A, Russell SJ (eds) *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann
5. Lee W, Stolfo SJ (1999) Combining knowledge discovery and knowledge engineering to build idss. In *Recent Advances in Intrusion Detection (RAID'99)*
6. Mukkamala S, Janoski GI, Sung AH (2000) Monitoring system security using neural networks and support vector machines. In Abraham A, Köppen M (eds) *Proceedings of the First International Workshop on Hybrid Intelligent Systems*, Advances in Soft Computing, pages 121–137. Physica-Verlag
7. Quinlan JR (1986) Induction of decision trees. *Machine Learning*, 1(1):81–106
8. Quinlan JR (1987) Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234
9. Schonlau M, DuMouchel W, Ju WH, Karr AF, Theus M, Vardi W (2001) Computer Intrusion: Detecting Masquerades. *Statistical Science*, 16:58–74
10. Stolfo SJ, Prodromidis AL, Tselepis S, Lee W, Fan DW, Chan PK (1997) JAM: Java agents for meta-learning over distributed databases. In Heckerman D, Mannila H, Pregibon D (eds) *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 74–81. AAAI Press
11. Teng H, Chen S, Lu S (1990) Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 278–284. IEEE Computer Society Press
12. Torres E (2003) *Sistema inmunológico para la detección de intrusos a nivel de protocolo HTTP*. PhD thesis, Pontificia Universidad Javeriana
13. Valdes A, Skinner K (2000) Adaptive, model-based monitoring for cyber attack detection. In Debar H, Mé L, Wu SF (eds) *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, RAID 2000*, volume 1907 of *Lecture Notes in Computer Science*, pages 80–92. Springer