VISUALISING INTRUSIONS: WATCHING THE WEBSERVER

Stefan Axelsson
Dept. of Computer Science, Chalmers University of Technology, Sweden
sax@cs.chalmers.se

Abstract Despite several years of intensive study, intrusion detection systems still suffer from a key deficiency: A high rate of false alarms. To counteract this, this paper proposes to visualise the state of the computer system such that the operator can determine whether a violation has taken place. To this end a very simple anomaly detection inspired log reduction scheme is combined with graph visualisation, and applied to the log of a webserver with the intent of detecting patterns of benign and malicious (or suspicious) accesses.

The combination proved to be effective. The visualisation of the output of the anomaly detection system counteracted its high rate of false alarms, while the anomaly based log reduction helped reduce the log data to manageable proportions. The visualisation was more successful in helping identifying benign accesses than malicious accesses. All the types of malicious accesses present in the log data were found.

1. INTRODUCTION

A significant problem with intrusion detection systems is the high number of false alarms [2, 8]. To address this *information visualisation* [4, 10] was applied to the problem of intrusion detection. The main problem with applying information visualisation to intrusion detection is the large amount of data that the user is faced with. To address this an anomaly detection inspired method was used to reduce the log to manageable proportions before utilising graph visualisation to understand the actual data. The hypothesis was that this would enable the user to benefit from the strengths of both visualisation—quickly making sense of medium size data sets, and anomaly detection—summarily discarding large amounts of uninteresting data, all the while avoiding the problems of visualisation having a limit to the amount of data that can reasonably

Y. Deswarte et al. (eds.), *Security and Protection in Information Processing Systems*

be handled, and anomaly detection having a high false alarm rate for decent detection rates.

An overview of this paper is as follows: First the experimental system is described in section 2. Then the anomaly based log reduction system is introduced in section 3 followed by a description of the visualisation and the report of the results from the visualisation and anomaly based log reduction in sections 4 and 5. The paper concludes with discussion, conclusion and future and related work in the remaining sections.

2. THE EXPERIMENTAL SYSTEM

For the experiment, a webserver access log was studied. It should be stressed that the primary interest is in experimenting with the effectiveness of the combination of visualisation and anomaly based log reduction, *not* in producing a realistic tool for usage in the field. Unfortunately there is a dearth of publicly available corpora useful for intrusion detection research. The most popular such corpora is the Lincoln Labs DARPA evaluation data. As it is export controlled it is unavailable to us.

The webserver under study serves a university computer science department. The server was running Apache version 1.3.26, and set to log according to the *common* log format. The log consists of a line based text file with each line representing a single HTTP access request. The *request* field i.e. the actual HTTP request sent to the server, is important as it is the central point of many attacks against a web server. The request field consists of the request method ('GET', 'HEAD', 'CONNECT', etc), followed by the *path* to the resource the client is requesting, and the method of access (e.g. 'HTTP 1.1'). The *path* in turn can be divided into components separated by certain reserved characters.

We studied the log for the month of November 2002, since it was believed that it would contain security relevant incidents, and we had access to later logs with which to compare the results. The access log contained ca. 1.2 million records. Selecting the actual request fields and removing duplicates ca. 220000 unique requests were identified. Because of their importance it is these unique requests that will be studied in the rest of the paper.

3. THE LOG REDUCTION SCHEME

Intrusion detection strategies can be roughly divided into two major categories, *signature based* and *anomaly based*. Anomaly based schemes rely on detecting *unusual* behaviour. Anomaly detection possesses the capability to detect *new attacks*, since it detects unusual patterns of activity, not illegal patterns per se. This capability comes at the cost of a much higher false alarm rate, to the point where the false alarms could drown all true alarms.

The log reduction scheme is based on descriptive statistics; in this case the frequencies with which events occur. This is in the same vein as seminal intrusion detection systems such as NIDES [1], though the approach here is simpler still. In order to classify the requests according to how unusual they are they are first cut up into components letting the reserved characters " ?:&=+$," separate the fields. For example a request such as 'GET /pub/index.html HTTP 1.1', is separated into the components 'GET', 'pub','index.html', 'HTTP' and '1.1'. The absolute frequencies of the fields as they appear in different unique request strings are counted.

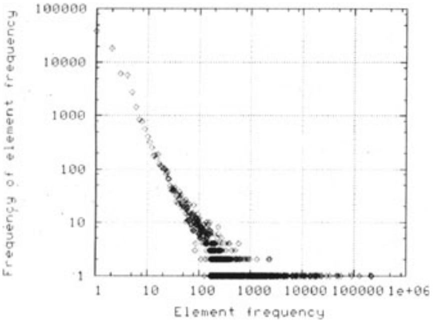

Figure 1. Frequencies of component frequencies

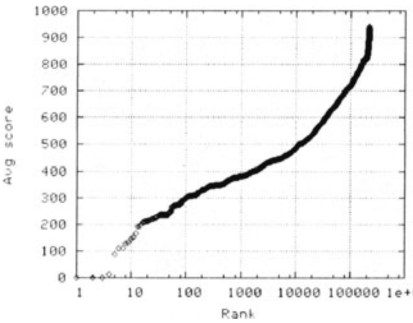

Figure 2. Requests sorted by lowest score

The request as a whole is scored by calculating the average of the absolute frequencies of the path components and hence requests consisting of unusual components have a low score, signifying that they are viewed as *anomalous*.. However, studying the *frequencies* of the *component frequencies* we see that a few high scoring elements (such as 'GET') could skew (i.e. drive up) the average. Therefore a *cutoff* is applied. Figure 1 lists the frequencies of the frequencies of the components. Studying the figure we see that a cutoff of 1000 seems reasonable since most of the activity appears to have died off by then. There are very few components with frequencies above 1000 and since they represent elements that are very common, they would tend to drown the lower frequency components we are interested in. Figure 2 plots the scores of the requests as a function of the ordering. The lowest scoring 5200 accesses are selected since that gives us a manageable amount of data to visualise.

4. VISUALISING THE LOWEST SCORING REQUESTS

The idea is to visualise the structure (and clusterings) of the various requests, the hypothesis being that differences in structure will enable the user to (relatively) quickly identify patterns of benign and malicious access. To accomplish this, the requests are cut into components as described in the previous section. The resulting components are visualised as a general graph where adjacent components in the request string are linked via directed edges in the graph. Using the same example as before: The request 'GET /pub/index.html HTTP 1.1', is cut up into nodes ('GET' etc.) with directed edges connecting 'GET' with 'pub', 'pub' with 'index.html' etc.

To visualise the resulting graph the graph visualisation tool *Tulip* was chosen.[1] Tulip has extensive features for interactive viewing and manipulation of general graphs. These aspects are unfortunately difficult to capture in writing (even with illustrations).[2]

To perform the actual detection the 5200 lowest scoring accesses is visualised in figure 3 [3] as a three dimensional general graph. The circular structure at the top of the graph that can be seen to reach almost all of the rest of the graph is the 'GET'-node. Note that the edges are not drawn as solid lines, since this would completely occlude the view.

At first figure 3 may look daunting, but closer scrutiny reveals several large features. Close to the center of the picture for example, we see a large double ring structure. Contrasting it with all other features, it looks rather unique, there is no other structure that looks similar (at least on this level), so we decide to investigate further.

Isolating the feature in question leads to figure 4. Following the links (which is somewhat difficult to do in the static display here) we learn of a loose structure that starts with either 'cgi-bin' or 'cgi-local' and progresses via 'recipient', 'subject' and then the unlikely looking random text strings. It turns out that these text strings are in fact recipient email addresses for *aol.com* and *hotmail.com* email users. The message to be mailed in many cases (but not all) purports to be from John Doe, and is simply "Is anybody out there?" So this particular access pattern seems

[1] Tulip is freely available under the GPL from 'http://www.tulip-software.org'.

[2] A more detailed version of this paper will be available on the author's website at the time of publication 'http://www.cs.chalmers.se/~sax'.

[3] The PDF-rendition of this graph may be clearer than a printed image.

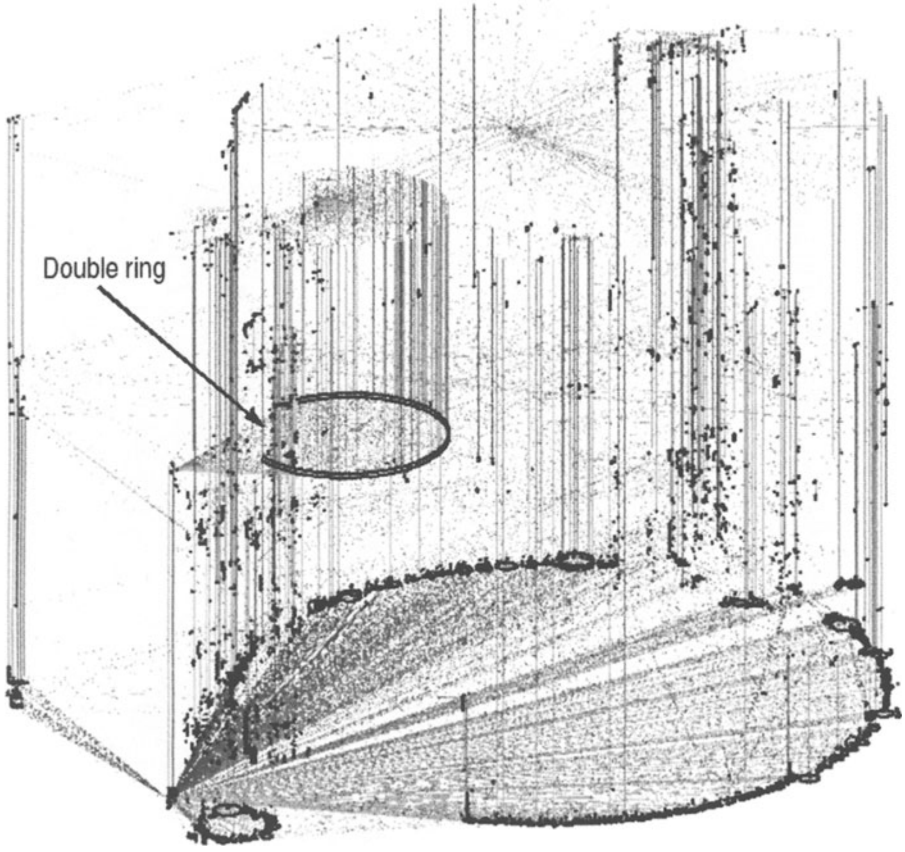

Figure 3. Graph of the lowest scoring requests

to be a *spam attack*, trying to use a misconfigured HTML to mail gateway that is commonly available. It was not active on the server however. The unlikely looking recipient names are probably automatically generated and the messages sent in the hope of eliciting a response and in doing so finding legitimate email addresses. Note that a tree visualisation would have been less powerful here, since there are two major (early in the request) parents of this particular pattern: 'cgi-bin' and 'cgi-local'. When visualised as a tree these branches would not have shared the latter features, even when they would have been the same.

We are usually not so fortunate as to identify attacks as we first lay our eyes on the graph. Instead we have to repeat the above detailed analysis. It so happens that all the other major features that are identifiable in figure 3 are uninteresting. However, they are all also much more regular

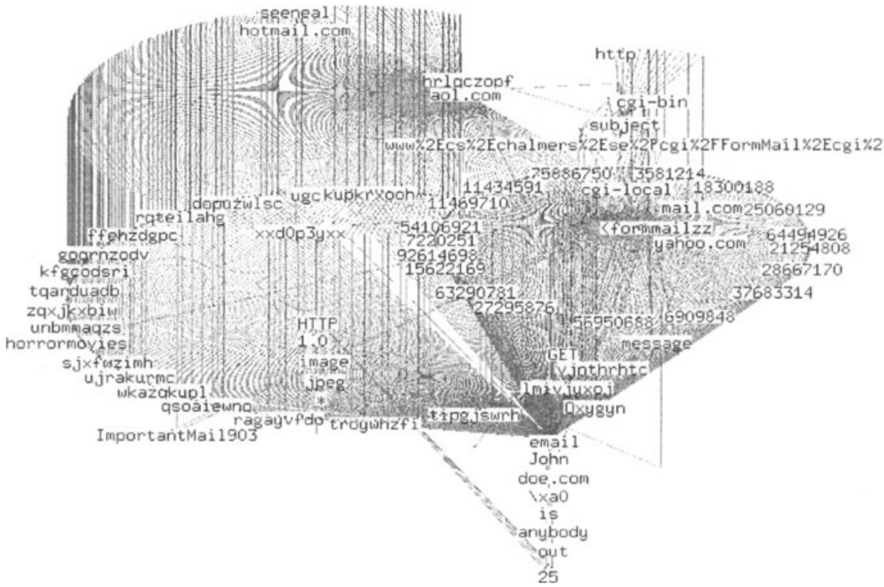

Figure 4. Zoom on feature (Spam attack in this case)

than the pattern we have just seen. This makes it possible for the user to eliminate many edges by eliminating fewer key parent nodes, slashing away what amounts to whole (improper) subtrees, and this fairly large graph can be withered down with a relatively modest amount of effort.

An example of this process is depicted in figure 5.[4] The accesses to files belonging to the user *carlsson* form a regular subtree that is easily identified as having nothing of a security relevant nature and hence easily discarded. This subgraph contains the data from 145 access requests, letting us discard some 3% of the reduced log in one fell swoop. Looking at the distribution of accesses in similar subgraphs, the twenty largest benign subgraphs (ranging from 213 to 59 access requests respectively) contain 42% of the access requests under scrutiny, and the first 180 benign subgraphs contain some 80% of all access requests in the reduced log. Hence most benign requests can be easily discarded. Unfortunately lack of space precludes more pictures of the details of this process than figure 5.

[4] For the purpose of illustration the corresponding access requests here have been isolated to make the graph clearer.

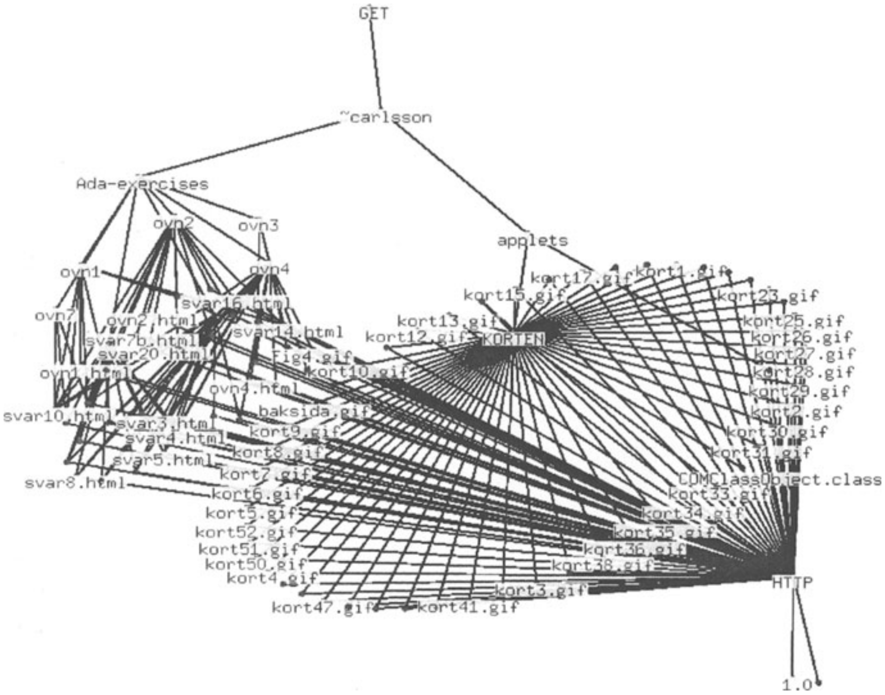

Figure 5. Zoom on feature (benign accesses forming a subgraph)

After about one to two hours the distilled requests that we cannot eliminate as being benign are arrived at, (since user experiments are yet to be performed on this method, a more precise estimate cannot be given with any certainty). These requests will be discussed in more detail in section 5.

5. DETAILED ANALYSIS OF THE FEATURES FOUND

The remaining accesses were classified into two categories, *suspect* and *intrusive.* The reason for using a *suspect* class is that since this is data from the field and the intentions of the entity submitting the request is not known, it is sometimes difficult to decide whether a request is the result of an intrusive process, the result of a flaw in the software that submitted it or a mistake by its user.

The *intrusive* class was further subdivided into seven different subclasses that correspond to metaclasses of the attacks that were observed:

Formmail attacks As already mentioned the server was subjected to a spam attack, in where spammers tried to exploit a commonly available web mail form to send unsolicited mail via the server.

Unicode attacks These are attacks against the Microsoft IIS web server, where the attacker tries to gain access to shells and scripts by providing a path argument that steps backward (up) in the file tree and then down into a system directory by tricking IIS into allowing e.g. '\..\' by escaping (as per the HTTP RFC) the offending backslash character in various ways. Many variations on the same basic scheme are present in the log data. Their visual structure is quite distinct from benign accesses.

Proxy attacks The attacker tried to access other web servers or IRC servers via the web server, hoping that it would be misconfigured to proxy such requests.

Pathaccess attacks These are more direct attempts to access command interpreters, cgi scripts or sensitive system files such as password files. A major class here is trying to access back doors known to be left behind by other successful system penetrations (mainly by worms). Configuration files of web applications, web store software for example were also targeted.

Cgi-bin attacks Attacks against cgi scripts that are commonly available and may contain security flaws.

Buffer overrun Only a few types of buffer overruns were found in the log data. All of these are known to be indicative of worms targeting the Microsoft IIS web server.

Misc This class contains seven accesses that probably are malicious in nature, but which do not fit in any of the previous classes. They are various probes using the *OPTIONS* request method, and a mixture of *GET* and *POST* request method calls that target the root of the file system.

The *intrusive* class (minus the *formmail* attack in figure 4) is depicted in figure 6. We directly see a few security relevant features (i.e. features that stand apart from benign accesses). One such feature is the 'system32'/'command.exe' tail in the lower middle of figure 6. This tail is common to many different branches from one of the roots of the graph and turns out to be a strong indicator of the *unicode* attack discussed above. Another peculiar cluster is at the top, second from the left, and it turns out to be instances of the meta class *proxy attacks.*

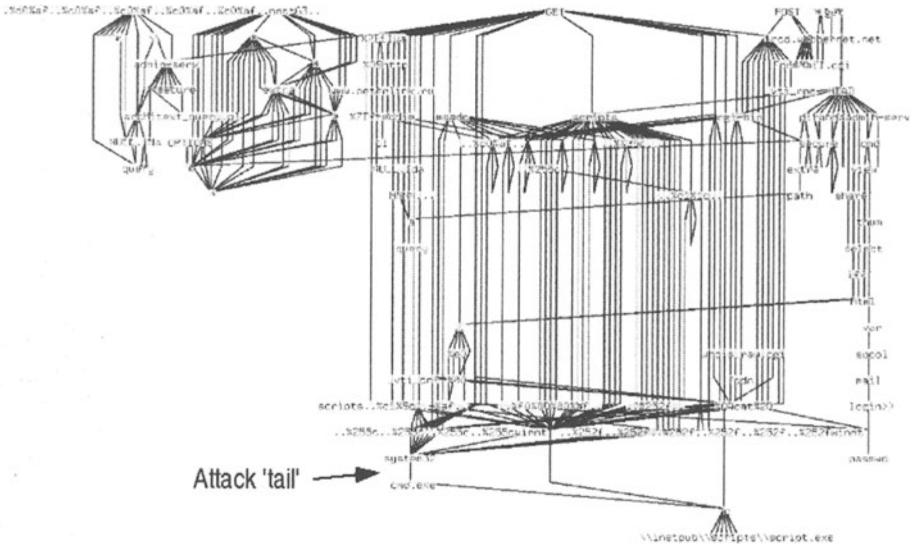

Figure 6. The remaining accesses deemed to be intrusion attempts.

As we can see from the graph there are many variations on the basic IIS encoding attack being tried against the server, although the graph of course does not list the actual attacks themselves. It would be interesting to study to what extent the graph predicts (all/most) possible attacks (i.e. by following one branch from the root to a leaf, do we get an executable attack that is not present in the log file proper). The overflow attacks are simple buffer overruns against fixed length buffers without adequate overflow protection, in our case these take the form of very long components (formed of a string beginning with repetitions of the character 'N' or 'A') followed by escaped shell code. Searching security sources we find that the first of these is characteristic of the Code Red worm.

Empirically in the data set studied here, the uninteresting access patterns are almost without fail very treelike in their appearance, with no common tails to speak of, while the opposite is true for the attacks. The reason seems to be that attacks that show some diversity nevertheless share common features that may come at any position in the request, while this is not true for normal accesses. In the case of unicode attacks we see it is the 'payload' i.e. the commands that the attacker wishes to execute that show many similarities, greater similarities in fact than the encoded path that leads to it. In the case of the spam attacks, it is the stereotype message delivered that is the key to the late similarities. It is conjectured that this would probably be difficult to avoid at least for the first kind of attack. There are only so many different commands to execute with the desired effect, and only so many file system paths to get to them, so there is bound to be a bottleneck (where the paths converge to a smaller set, and the command set starts), giving an hour glass shape. On the other hand the access to static legitimate web pages is highly tree-like in nature, and hence does not elicit the same hour glass shape. Cgi-bin arguments show more variation, but they do not in our experience contain the same clearly identifiable tails as the attack patterns. In the case of the scripts for collecting student opinion of courses taken for example, the text of the messages, while sharing many words between different opinions, still display much more variation, giving an almost random appearance, not the typical hour glass shape of the suspect patterns. However, not all attack patterns show this hour glass shape, the attacks with very little variation in the attack types does not provide enough data for a pattern to emerge. It should be noted that being subject to a large number of different variations of the same kind of attack *increases* the possibility of the attack being detected using the scheme presented here, since there is more structure apparent. The opposite is typically true of signature based intrusion detection systems that have been programmed to detect one type of attack (or at least a smaller range of attacks).

However, despite the structure apparent in the attacks, comparing e.g. figure 5 to figure 6 indicates that the visualisation method is better suited to *eliminate* benign requests than to *detect* malicious requests. Which is just as well given that there are many more benign requests than malicious ones; and in fact the *false alarms* suppression capability of an intrusion detection system determines its effectiveness [2].

6. EFFECTIVENESS OF THE LOG REDUCTION SCHEME

In order to evaluate the effectiveness of the log reduction scheme we analysed the entire 220000 unique web requests by hand. This proved to be an extremely tedious task, the author spent a week classifying and investigating the data.

The access requests were classified into the three categories: *normal*, *intrusive* and *suspect* as before. Furthermore the *intrusive* class was divided into the seven subclasses discussed earlier. This further subdivision benefits the investigation of the effectiveness of the log reduction mechanism since it avoids reporting results that might indicate an overall satisfactory detection rate, but on closer study turn out to be lacking in detection capability in any of the seven areas[5]. A summary of how many of the different classes of attacks the log reduction mechanism detected can be found in table 1.

Instances of the class *suspect* are not considered false alarms when they occur in the output of the log reduction tool, but on the other hand will not be considered missed attacks when they do not.

Attack	*Total*	*Alarms*	*Detection rate (%)*
Formmail	285	282	99
Unicode	79	79	100
Proxy	9	5	56
Pathaccess	71	16	23
Cgi-bin	219	17	8
Buffer overrun	3	2	66
Miscellaneous	7	2	29
All attacks	673	403	60

Table 1. Detection rates of the log reduction mechanism.

As we can see in table 1, the log reduction mechanism faired well, it managed to preserve evidence of all seven classes of attacks in the reduced log. In light of these results the author would not hesitate to claim the log reduction mechanism a success from the point of view of detection rate, even though it did not do spectacularly well in all classes, especially in the classes that were most similar to normal traffic, which is only to be expected.

The discussion of false alarm rates is complicated by the fact that anomaly detection is not actually done, but log reduction. The difference is that an anomaly detection system reports on the *absolute* abnormality of a request, while the log reduction system reports on the *relative* abnormality of a request. The corresponding anomaly detection system would produce a variable number of alarms indicating the level of intrusive activity, but the log reduction scheme will always report the same number of 'alarms' regardless, as a suitable number of 'alarms' are selected to perform visualisation on. Thus the notion of *false alarm rate* is not well defined in this context. Two papers discuss the issue of alarms, and false alarms in intrusion detection systems in more detail [2, 8].

Table 2 below lists the absolute values of the number of *attacks*, *normal traffic* and *suspect traffic* that are evident in the whole data set and

[5] This might well have been the case since the distribution of different accesses in each of the seven subclasses turns out to be highly skewed and so doing well in one class would skew the overall result.

in the reduced log. As might have been expected the number of benign accesses in the reduced log is quite high. Circa 86% of the reduced log contains benign traffic. (Note that this number would not correspond to the *false alarm rate*, but rather the *Bayesian false alarm rate* as defined in [2]. The true false alarm rate would be even higher.) This is acceptable since the number of accesses are kept within reasonable limits for the chosen visualisation method and the log reduction mechanism has a sufficiently high detection rate.

Access type	*All accesses*	*Alarms*
Attacks	673	403
Normal traffic	215504	4499
Suspect	115	28
Total	220000	5200

Table 2. Summary of the true and false alarms of the log reduction mechanism.

Note that it is pointless to compare the results from the system presented here with that of popular signature based systems such as *Snort*, since these rely on previous external knowledge of intrusions. There is unfortunately a dearth of suitable publicly available anomaly detection systems with which to compare the results, though that would be more useful.

7. DISCUSSION

Comparing the presented method with spending the same amount of time going through a false alarm list from an intrusion detection system, the user has been spent time actually learning about the type of traffic the webserver sees, knowledge that can be used to make the site run better/smoother. This is not generally the case when watching the output of an intrusion detection system.

This generalized knowledge—patterns that are distinct from benign traffic have been found, no benign traffic contains the kind of encoding that the encoding attacks does—can of course be used to program a signature based intrusion detection system. The author conjectures it would work well since it is known what elements of these intrusions that are unique to attacks and do not occur in normal traffic to the specific site. When performing intrusion detection the best results are achieved when having a model of both the normal and the intrusive traffic [8]. At the very least this knowledge can be used when tuning a signature based intrusion detection system, something which is always necessary.

The anomaly based log reduction system faired well, and it is believed that it would furthermore be difficult to try to circumvent it by injecting similar fields in other fake access requests to try and drive up the frequencies of the interesting fields. This is because such accesses would by their very nature not be successful either as attacks, or as access requests. Taking the result code (i.e. 404) into account when performing the log reduction would eliminate these fake accesses. This would perhaps come at the cost of a decreased detection of *attempted* intrusions. This could be addressed by looking at access requests that were either successful, or unsuccessful, or similar access requests, some of which were successful and some of which were not. As any 'chaff' access requests by their very nature must be unsuccessful there is a difference between them and other access requests (either benign or malicious) that can be put to work for detection.

On a different tack, the cost of going through the unique requests can be amortised. Studying the logs for the months following November, i.e. December through February, they contain on the order of the same number of unique requests in themselves, viewed in isolation. However, amortised there are 200000 previously unseen requests in Nov., 87000 in Dec., 66000 in Jan., and only 40000 in Feb.

A disadvantage with this approach compared to that of automated intrusion detection systems is that the detection is not necessarily real time or near real time. Especially if is decided to visualise the log in batches of one month at a time. Of course, nothing prevents the operator from performing the visualisation more often.

8. FUTURE WORK

The discussion has been limited to the *types* of different attacks seen in the web log (the same request string could emanate from many different sources). No attempt has been made to correlate the actual attacks with each other, or cluster the same attacks originating from different sources to try and identify the entity behind the attacks. Methods to do so (even visual methods [3]) already exist.

It would also be interesting to try the (few) other data sets such as the KDD'99 data set and to see if the structural differences found here carries over to other types of data.

It would be interesting to devise methods of evasion as noted above, and implement the suggested improvements to the log reduction to thwart them. A lack of time and space has prevented us from doing so here.

The intention is also to perform user experiments. These are more difficult than one might at first think, since training on the specific tool often is very effective for the outcome, and the task to be performed is complex and demands some skill. This makes the experiment prohibitively costly.

9. RELATED WORK

The idea of bringing visualisation to intrusion detection was first published in 1998, and a body of work has started to collect.

A small subfield (see e.g. [9, 5]) of anomaly detection and visualisation has arisen through the application of Kohonen maps to intrusion detection which itself is a visual representation of an underlying neural network model. These systems all build some neural network model of network traffic or host data and then present the resulting two dimensional scatter plot to the user. The interpretation of the map is known to be difficult [6].

These approaches all differ from the one presented here in that none perform log reduction similar to the scheme presented in this paper, but rather anomaly based intrusion detection, and that neither paper argue the effect the visual presentation has on the understanding of the security result by the user of the system. Furthermore they all use the raw data as input, not divided into categories (i.e. *accesses* in this case), and in doing so detect instances of attacks, but not necessarily categories of attacks as done here.

A quick survey of the available commercial intrusion detection systems was also made. Only three systems uses any degree of visualisation in our sense of the word. Two are uninteresting from our perspective but the remaining, *CA Network Forensics*: 'http://www.silentrunner.com', uses N-gram clustering followed by a three dimensional visual display of the clusters. On the surface the visual representation of the data in the clusters is similar to the one presented here (i.e. a general 3D network) but while the graphs may look similar they express very different relations. There is no discussion as to the interpretation of these graphs and the underlying structure of the data is not allowed to influence the visualisation.

The main difference between these works and the one presented here is that the one presented here combine anomaly based log reduction techniques with visualisation techniques, while they rely on clustering techniques combined with a visual display of the clusters, i.e. they visualise the result of the anomaly detection method, while the *data* that is singled out by the log reduction method is visualised here.

Theo et. al. [11] visualise communication between pairs of routers on the Internet using the BGP (Border Gateway Protocol) routing protocol. Their choice of problem and visualisation techniques are different from the one presented here, and they do not delve as deeply into the analysis of the security problems they detect (they are not as clearly *security problems*), but they do visualise a greater amount of data more compactly than done here and still manage to detect anomalies in the BGP traffic.

There has not been much research into anomaly detection of web accesses besides that by Kruegel et.al. [7]. They develop (as is done here) ad hoc statistical methods for detecting anomalous request strings. Their model is much more complex than the one presented here, taking many more parameters into account while only one (the element frequency) is taken into account here. As a result—as far as false alarm rates can be compared between a detector and a log reducer—they are rewarded with a false alarm rate about a factor of forty lower than the one reported here (and possibly a detector that's more resistant to evasion attempts). Even so the authors report a problem with handling even this level of false alarms, while the visualisation method presented here enables the user to quickly discard the uninteresting entries.

10. CONCLUSIONS

In summary, the hypothesis that the combination of anomaly based log reduction and visualisation would provide us with the benefits of both approaches while counteracting the drawbacks was supported. Furthermore the anomaly based log reduction system could indeed be very simple and still successfully serve as a front end to the visualisation system. The hypothesis that visualising the structure of the requests strings themselves cut into components would enable the operator to discard benign accesses with relative ease was supported. There was less evidence for the corresponding hypothesis: That one could just as easily identify malicious patterns. A few meta classes of attacks did exhibit features that set them apart from the benign traffic, but the others did not to a significant degree.

The presented method is relatively time efficient, and the operator learns about the usage of the website. Notably unusual but benign (often dynamic) traffic that is more varied and hence more prone to misclassification is studied in more detail.

The work invested in parring down the graph can be amortized over subsequent investigations, where the webserver logs for the following months contain less and less new traffic, and hence can be visualised more quickly, especially if one remembers what accesses were seen previously and why it was decided to discard them as uninteresting.

References

[1] D Anderson, T Frivold, and A Valdes. Next-generation intrusion-detection expert system (NIDES). Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, CA 94025-3493, USA, May 1995.

[2] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

[3] Stefan Axelsson. Visualization for intrusion detection: Hooking the worm. In *In the proceedings of the 8th European Symposium on Research in Computer Security (ESORICS 2003)*, volume 2808 of *LNCS*, Gjøvik, Norway, 13–15 October 2003. Springer Verlag.

[4] Stuart K. Card, Jock D. MacKinlay, and Ben Shneiderman. *Readings in Information Visualization—Using Vision to Think.* Series in Interactive Technologies. Morgan Kaufmann, Morgan Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Fransisco, CA 94104-3205, USA, first edition, 1999. ISBN 1-55860-533-9.

[5] Luc Girardin and Dominique Brodbeck. A visual approach for monitoring logs. In *In the Proceedings of the 12th Systems Administration Conference (LISA '98)*, pages 299–308, Boston, Massachusetts, USA, 6–11 December 1998. The USENIX Association.

[6] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences.* Springer Verlag, Third edition, 2001. ISBN 3–540–67921–9, ISSN 0720–678X.

[7] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03)*, pages 251–261, Washington DC, USA, October 2003. ACM Press.

[8] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*, Oakland, California, USA, 14–16 May 2001. IEEE.

[9] P. Lichodzijewski, A.N. Zincir-Heywood, and Heywood M.I. Host-based intrusion detection using self-organizing maps. In *In the proceedings of the IEEE International Joint Conference on Neural Networks.* IEEE, May 2002.

[10] Robert Spence. *Information Visualization.* ACM Press Books, Pearson education ltd., Edinburgh Gate, Harlow, Essex CM20 2JE, England, first edition, 2001. ISBN 0-201-59626-1.

[11] Soon Tee Teoh, Kwan-Liu Ma, S. Felix Wu, and Xiaoliang Zhao. Case Study: Interactive Visualization for Internet Security. In *Proceedings of IEEE Visualization 2002*, The Boston Park Plaza hotel, Boston, Massachusetts, USA, 27 October to 1 November 2002. IEEE Computer society.