Paper

Application of bioinformatics methods to recognition of network threats

Adam Kozakiewicz, Anna Felkner, Piotr Kijewski, and Tomasz Jordan Kruk

Abstract— Bioinformatics is a large group of methods used in biology, mostly for analysis of gene sequences. The algorithms developed for this task have recently found a new application in network threat detection. This paper is an introduction to this area of research, presenting a survey of bioinformatics methods applied to this task, outlining the individual tasks and methods used to solve them. It is argued that the early conclusion that such methods are ineffective against polymorphic attacks is in fact too pessimistic.

Keywords— network threat analysis, sequence alignment, edit distance, bioinformatics.

1. Introduction

When biologists discover a new gene, its function is not always apparent. The usual approach is to compare its structure with genes, whose function has already been identified. Comparison (so-called *alignment*) of biological sequences is a basis for bioinformatics, a science focused on theories, algorithms, computational techniques and statistical methods, with the goal of solving problems of biological data analysis [1]. Bioinformatics draws inspiration from many other branches of science and the techniques it provides often have interesting applications outside of biology – including automatic voice and handwriting recognition, but also in computing systems security. This paper focuses on methods of defining the similarity between biological sequences and shows, how similar methods can be applied to the problem of recognition and characterization of computer network threats.

2. Sequence alignment

Sequence alignment is a tool used in bioinformatics to define and visualize a measure of similarity between DNA or protein sequences.

Definition 1 (from [2]) : A (global) *alignment* of two strings S_1 and S_2 is obtained by first inserting chosen spaces (or dashes), either into or at the ends of S_1 and S_2, and then placing the two resulting strings one above the other so that every character or space in either string is opposite a unique character or a unique space in the other string.

For example, in the alignment

```
c  a  c  _  d  b  d
c  a  w  x  _  b  _
```

of strings *cacdbd* and *cawxb*, character *c* is mismatched with *w*, both *d's* and the *x* are opposite spaces, and all other characters are in matches.

Definition 2 (from [2]) : A global *multiple alignment* of $k > 2$ strings $S = S_1, S_2, ..., S_k$ is a natural generalization of alignment for two strings. Chosen spaces are inserted into (or at either end of) each of the *k* strings so that the resulting strings have the same length, defined to be *l*. Then the strings are arrayed in *k* rows of *l* columns each, so that each character and space of each string is in a unique column.

Alignment is necessary, since evolutionary processes introduce mutations in the DNA and biologists do not know, whether *n*th symbol in one sequence indeed corresponds to the *n*th symbol of the other sequence – a shift is probable.

Definition 3 (from [2]) : The *edit distance* between two strings is defined as the minimum number of edit operations – insertions, deletions, and substitutions – needed to transform the first string into the second. For emphasis, note that matches are not counted.

Edit distance is sometimes referred to as *Levenshtein distance* in recognition of the paper [3] by V. Levenshtein where edit distance was probably first discussed.

Sequence alignment is a generalization of an intuitive approach to analysis of similarity between sequences, based on searching for the longest common subsequence (LCS). The LCS is found by inserting gaps in the two sequences, so that they can be aligned with a maximum possible number of matching characters. This operation uses a simple *scoring function*: "+1 for a matching character, 0 otherwise." In real biological applications more complicated *scoring matrices* are used, assigning more points to matching known functional biological sequences, giving points for aligning characters which do not strictly match, but are similar from the point of view of their biological function, and also subtracting points for non-matching characters and gaps in unexpected places. The final score is usually simply a sum of points for all pairs of characters. The alignment is therefore optimal, when the similarity is greatest, resulting in the highest score with the selected scoring matrix [4].

The optimal path leading to the best alignment is found using dynamic programming algorithms. The most well-known dynamic programming algorithm used in bioinformatics for this task is the Needleman-Wunsch algorithm. The goal is to find the best global alignment of two se-

quences of characters. Global alignment answers the question, how similar are the two compared sequences along their entire length. Usually the compared sequences are of similar length, however in some cases the sequences are very different, except for some shorter, similar subsequences. Sometimes it is also necessary to compare sequences of very different lengths. In this case a local alignment algorithm is better. An example of such algorithm is the Smith-Waterman algorithm, a modification of the Needleman-Wunsch algorithm. Given a scoring function, the algorithm is designed to find the most similar pair of subsequences of both sequences. In the local alignment problem the final score is computed only on the pair of subsequences, omitting the rest of the characters. The difference between local and global alignment is illustrated below [5].
Global alignment:

```
--AGATCCGGATGGT--GTGACATGCGAT--AAG--AGGCGTT
    ||| |  |  |  ||||| ||||||  |||  |  | ||
GTCCATCTG--TCTTGGGTGAC-TGCGATACAAGTTA--CCTT
```

Local alignment:

```
--AGATCCGGATGGT--[GTGACATGCGATA]--AG--AGGCGTT
                  ||||| |||||||
GTCCATCTG--TCTTGG[GTGAC-TGCGATA]CAAGTTA--CCTT
```

3. Sequence alignment and network threat recognition

Sequence alignment is a useful tool for network threat recognition in intrusion detection systems, automated threat signature generators and in malware analysis. To recognize a threat it is necessary to compare the new, observed behavior with previously identified or exemplary malicious behavior. The comparison is more useful, if it is not strict – this facilitates the detection of variants of attacks (behaviors) or attempts at masking the attack. Just like in bioinformatics problems, the main task is to find regularities, repeating similar sequences in large datasets.

4. Anomaly detection

One of the first applications of sequence alignment to intrusion detection is mentioned in [5] – with explicit connection to bioinformatics. The authors focused on the problem of detection of masquerading attempts, using logs of the `acct` tool in Unix systems as input data. Masquerading detection involves a *user signature* – a sequence of commands collected from the user, compared with the current session of this user. The main assumption is that an intruder using another user's account will behave in a different way than the rightful owner of the account, and that this difference of issued commands should be detectable. The paper proposes a method of aligning sequences of commands from the current session with the user signature using a modified version of the Smith-Waterman algorithm. The result of the alignment, using a proposed scoring function, is used to detect an intrusion. In the opinion of the authors a local alignment would not be sufficiently effective, as a lot of potentially interesting data would be ignored. Therefore, the authors have used a *semi-global* alignment. In this type of alignment only the suffixes or prefixes of compared sequences are aligned. The scoring function rewards matching commands, but does not penalize the existence of large, non-matching parts of the signature. As in every anomaly detection method, an arbitrary threshold must be chosen to separate a suspected intrusion from a normal, but slightly atypical session of the original user – this threshold was found empirically. The best experimental results show a 75.8% intrusion detection level with 7.7% of false positives. A full description of the algorithm, the scoring function, data preparation method and analysis of results can be found in [5].
Another method of intrusion detection, popular in the literature but rarely used on a wide scale is system call monitoring – the system calls of processes are monitored and compared to typical behavior of a given type of processes. Differences in behavior could indicate a successful attack on the application, resulting in execution of potentially malicious code. A recently proposed extension of this idea is based on *evolutionary distance* between sequences, defined as the sum of costs of substitutions, deletions and insertions. Instead of creating a model of the behavior of the monitored process, a "replica" of the process is created and executed in parallel. A difference in behavior of the two processes may be a symptom of an attack. Since an effect of the same attack on two identical processes on identical platforms must, by definition, be the same, diversification of replicas is necessary. Thus, good candidates for a replica are: the same process running on a different platform (e.g., Windows instead of Linux), or even a different process with the same functionality (e.g., a different WWW server) on a different platform. The authors of the idea assume that even though the processes will use different system calls, the function of those calls will be similar. It is possible to correlate different system calls from different processes/platforms. A description of the method of computing the behavioral distance between processes and of the experimental results can be found in the paper [6]. In the next paper [7] the authors used hidden Markov models for this task.

5. Threat signature generation

Bioinformatics are much more often mentioned in the literature on network threats in the context of threat signature generation systems. This area of research has gained a lot of attention in the recent years. New, unknown threats appear very often. They are initially not recognized by the traditional intrusion detection systems based on threat signatures, since a signature has not been created yet. In this case a very useful tool is an automatic system, capable of recognizing a new threat and generating its signature, preferably without human intervention.

The first system to automatically generate threat signatures was `honeycomb` [8], a plug-in for `honeyd`. While the system itself was not based on bioinformatics methods, it did use some algorithms for detection of repeating similar sequences. The system applied the longest common substring[1] algorithm to find common sequences of bytes in different packets sent to the system. As the system was a part of a honeypot, all incoming packets were by definition suspected to be part of an attack. Unfortunately, the system did not scale well in real honeypot networks, it generated a lot of repeating signatures and was completely useless against polymorphic attacks. Additionally, lack of implemented signature management methods meant that with time it was difficult to tell, which signatures (and attacks) are indeed new.

`Honeycomb` had many successors, using different methods to recognize repeating sequences in the data stream, using them as the basis for signature generation. However, more advanced bioinformatics algorithms were not proposed until polymorphic attacks were targeted. In a polymorphic attack there are, by definition, few constant substrings (subsequences without gaps), common among all instances of the attack. Furthermore, the longest such substring, if found, is not necessarily the best sequence describing the attack.

For many years identification of a polymorphic attack using signatures expressed as subsequences of the attack was thought impossible. Signature-based intrusion detection systems were expected to disappear soon. However, in paper [9] it was shown, that every polymorphic attack must contain constant, repeating values, allowing the attack to successfully exploit a given vulnerability. Some constants are also required to use a given protocol to communicate with the attacked application. Description of such an attack is, therefore, possible, although difficult – a good description of the attack is neither a single common substring, nor the longest common subsequence, which might contain too many random individual characters. A local alignment is necessary to find a common region in all variants of a polymorphic attack. This approach was suggested in the `polygraph` system. It is a signature generation system, using information from another system to identify suspect flows. Using a set of such flows a signature is created as a set of short separate character sequences. For example, a signature for the Apache-Knacker exploit was as follows (expressed as a regular expression):

```
GET .* HTTP/1.1\r\n.*:.* \r\nHost:.* \r\n

.*:.*\r\nHost:.*\xFF\xBF.*\r\n
```

To find common characters for the flows the authors used a modified version of the Smith-Waterman algorithm. The modification included rewarding continuous alignment, since such signatures are less likely to cause false positives. Groups of characters were rewarded, while gaps were penalized, where gaps are not only the maximal length of subsequences matched with spaces, but also the maximal length of subsequences of non-matching characters. The penalties were selected so that character sequences were more likely to be aligned if their grouping is typical for a given protocol. In practice this would mean that different scoring functions for different protocols should be developed.

In the experiment carried out by its authors, the system was tested on three real exploits – two for `httpd` servers (the Apache-Knacker exploit and the ATPhttpd exploit) and one for BIND server (the BIND-TSIG exploit). Clet, a well known tool for polymorphic attack generation was used. It was found, that Clet had many weaknesses – in each variant of the exploit many constant sequences were found. Since the goal of the experiment was to test the system with the assumption of nearly perfect polymorphism, the code of the exploit was manually changed using random values, leaving the sequences necessary for its functioning intact. The signature generator based on the modified Smith-Waterman algorithm produced the correct signature in all tests, giving 0.0008% false positives for Apache-Knacker, and 0% for the BIND-TSIG exploit – verified against a test pool of "proper" traffic. Results for the ATPhttpd exploit were not published. Only 3 samples of the exploit were necessary to reach such a high level of precision. The generated signatures can be used in many modern intrusion detection systems like `snort`.

Another approach to polymorphic worm detection was used in [10]. Like in the previous case, common regions were searched for – using *Gibbs sampling* and creating signatures based on the frequencies of individual characters. Gibbs sampling is also used in bioinformatics to find *motifs* – unchanged by evolution regions in protein sequences.

6. Honeypot development

Bioinformatics methods can be of great value for honeypot developers. Honeypots are often the main source of information on new threats and a basis for early warning systems. The best honeypots are real systems with real applications. However, their installation and management is complicated and time consuming. This led to the creation of honeypots like `honeyd` and `nepenthes` [11], emulating the attacked services. They are easy to deploy and manage, even on a large scale, because the attacks on emulated services are – by definition – never successful, so the honeypots themselves are not infected. The downside of emulation are its limitations – it is never perfect (bug-for-bug compatibility is difficult to attain, especially for unknown bugs), and a honeypot is only as good as the emulation. The problem is how to create new emulation modules for services and their vulnerabilities in a fast and easy way. Automatic tools have been proposed – including `ScriptGen`,

[1] A *different* term than longest common subsequence (LCS) – it is by definition continuous, while the LCS may contain gaps. The resulting alignment is therefore different – the longest common substring usually is not simply the longest continuous part of the LCS.

which can generate new honeyd scripts and aims to add the ability to generate nepenthes modules. It functions in three steps:

- running a real system as honeypot, registering incoming traffic;
- traffic analysis without knowing the semantics of the observed protocols, by aligning many sequences of requests and answers and building a state automata based on the result of this analysis;
- creating a honeyd script based on the state automata.

A full description of the methods can be found in papers [12] and [13].

7. Summary

Bioinformatics methods are a promising approach to the problem of network threat detection and recognition. However, in most cases their only application are tests in laboratories. It seems that their most promising application from the practical point of view would be the generation of signatures for polymorphic attacks, more general signatures with a low level of false positives, creation of tools for management of automatically generated signatures and for extending the capabilities of honeypots (in tools like ScriptGen). An important conclusion is that bioinformatics methods have shown that the popular opinion that polymorphic attacks will make description and detection based on signatures of characteristic sequences obsolete is wrong.

References

[1] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*. Cambridge: MIT Press, 2004.

[2] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge: Cambridge University Press, 1997.

[3] V. I. Levenshtein, "Binary codes capable of correcting insertions and reversals", *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[4] P. Kijewski, "Zastosowanie metod bioinformatyki do rozpoznawania zagrożeń sieciowych", in *SECURE 2006 Bezpieczeństwo – czas na przełom*, Warsaw, Poland, 2006 (in Polish).

[5] S. Coull, J. Branch, B. Szymański, and E. Breimer, "Intrusion detection: a bioinformatics approach", in *Proc. 19th Ann. Comput. Secur. Appl. Conf.*, Washington, USA, 2003.

[6] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance for intrusion detection", in *Proc. 8th Int. Symp. Recent Adv. Intrus. Detect. RAID 2005*, Seattle, USA, 2005.

[7] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance measurement using hidden Markov models", in *Proc. 9th Int. Symp. Recent Adv. Intrus. Detect. RAID 2006*, Hamburg, Germany, 2006.

[8] C. Kreibich and J. Crowcroft, "Honeycomb – creating intrusion detection signatures using honeypots", in *Proc. 2nd Worksh. Hot Top. Netw. Hotnets II. ACM SIGCOMM*, Boston, USA, 2003.

[9] J. Newsome, B. Karp, and D. Song, "Polygraph – automatically generating signatures for polymorphic worms", in *Proc. IEEE Symp. Secur. Priv. SP 2005*, Washington, USA, 2005, pp. 226–241.

[10] Y. Tang and S. Chen, "Defending against Internet worms: a signature-based approach", in *Proc. 24th Ann. Conf. IEEE INFOCOM 2005*, Miami, USA, 2005.

[11] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The nepenthes platform: an efficient approach to collect malware", in *Proc. 9th Int. Symp. Recent Adv. Intrus. Detect. RAID 2006*, Hamburg, Germany, 2006.

[12] C. Leita, K. Mermoud, and M. Dacier, "ScriptGen: an automated script generation tool for honeyd", in *Proc. 21st Ann. Comput. Secur. Appl. Conf. ACSAC 2005*, Tucson, USA, 2005.

[13] C. Leita, M. Dacier, and F. Massicotte, "Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots", in *Proc. 9th Int. Symp. Recent Adv. Intrus. Detect. RAID 2006*, Hamburg, Germany, 2006.

Adam Kozakiewicz graduated from the Faculty of Electronics and Information Technology of Warsaw University of Technology, Poland. Currently he works as a Research Associate at Systems and Information Security Methods Team in NASK Research Division and an Assistant at the Institute of Control and Computation Engineering at the Warsaw University of Technology, where he awaits the defense of his Ph.D. thesis in telecommunications. His main scientific interests includes security of information systems, parallel computation, optimization methods and network traffic modeling and control.
e-mail: adam.kozakiewicz@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Anna Felkner graduated from the Faculty of Computer Science of Białystok Technical University, Poland. At present she works as a Research Assistant at Systems and Information Security Methods Team in NASK Research Division. She is a Ph.D. student in the Institute of Control and Computation Engineering at the Warsaw University of Technology. Her main scientific interests concerns the security of information systems.
e-mail: anna.felkner@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Piotr Kijewski holds an M.Sc. degree in telecommunications from the Warsaw University of Technology. He works for NASK since 2002, as an IT Security Specialist in the CERT Polska team. His main interests in the computer and network security include intrusion detection, honeynets and network forensics. He is the leader of the team that designed, implemented and deployed the nation-wide early warning system based on honeypots in Poland. He has taken part in EU funded security projects (eCSIRT.net, SpotSpam). He is the author of over two dozen papers on security topics, and has spoken on conferences both inside and outside of Poland.

e-mail: piotr.kijewski@cert.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Tomasz Jordan Kruk graduated in computer science from Gdańsk University of Technology (M.Sc., 1994) and Warsaw University of Technology (Ph.D., 1999). From 1994 to 2001 he was responsible for supercomputers and services administration in Warsaw University of Technology Computing Centre. Since 1999 he works as an Associate Professor (Adjunct) at the Warsaw University of Technology. Since 2001 also Associate Professor (Adjunct) at Research and Academic Computer Network (NASK). His main areas of interests cover IT security, operating systems and distributed/cluster systems. Currently he heads the Systems and Information Security Methods Team in NASK Research Division.

e-mail: T.Kruk@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland