Network Protocol Analysis using Bioinformatics Algorithms

Marshall A. Beddoe
Marshall_Beddoe@McAfee.com

ABSTRACT

Network protocol analysis is currently performed by hand using only intuition and a protocol analyzer tool such as tcpdump [6] or Ethereal [7]. This paper presents Protocol Informatics, a method for automating network protocol reverse engineering by utilizing algorithms found in the bioinformatics field. In order to determine fields in protocol packets, samples are aligned using multiple string alignment algorithms and their consensus sequences are analyzed to understand the beginning and the end of fields in the packet. For illustrative purposes, this paper uses the common HTTP protocol as an example of a "blackbox" protocol as it is comprehensive and easy to follow. While HTTP is a plaintext protocol, the same principle applies to any byte stream that contains structure.

Keywords: Network Protocol Analysis, Bioinformatics, Sequence Alignment.

1. INTRODUCTION

The objective of protocol analysis is to determine the location and lengths of fields within protocol packets. Reverse engineering these packet formats allows one to understand the structure of requests and responses. The challenge today with protocol analysis is that there are no automated methodologies. Packets that contain only 8-32 bit integer fields are trivial to reverse engineer by hand, however, the effort dramatically increases when variable length fields are introduced into a packet format. While attempting to solve this problem by using edit distances and entropy measurements, it became apparent that this problem had already been solved in the realm of bioinformatics.

Bioinformatics is the use of informational and mathematical techniques to solve biological problems such as sequence alignment. This involves the processing of large amounts of structured, yet complex data. Whereas in bioinformatics the purpose is to identify specific genes that produce proteins, the purpose of protocol analysis is to identify the location and purpose of fields in packet formats. This paper describes how to apply string alignment algorithms and phylogeny concepts to protocol analysis in order to automatically identify fields in network packets.

The value gained from protocol analysis ranges from general understanding to security implications. In the realm of computer security, this information is important because it allows client-server applications to be assessed for vulnerabilities in a "blackbox" manner by crafting packets that may trigger buffer overflows within the application.

The rest of this paper is organized as follows. Section 2 contains the details for sequence alignment algorithms. Section 3 describes multiple alignment algorithms using phylogenetic trees. Section 4 presents an evaluation of these algorithms applied to the HTTP protocol and also explains the value gained from automated field identification. We conclude in Section 5.

2. SEQUENCE ALIGNMENT

Sequence alignment is used in biology to understand the relationship between two sequences of genetic information, such as DNA or amino acids. There are databases of genetic information which are processed by string alignment algorithms to better understand the relationship between species and to also determine the location of specific genes. For protocol analysis, the concept is similar. Compare a sample to a database of packets belonging to a specific protocol. This allows one to understand the type of packet in the overall protocol scheme and also allows one to determine the location and size of the fields in each individual packet.

The concept of sequence alignment is simply to align two sequences to the same length by inserting gaps when necessary. Starting with an example, the sequence "GET / HTTP/1.0" and "GET /index.html HTTP/1.0" is shown in Figure 1.

```
1 GET /index.html HTTP/1.0
2 GET /__________ HTTP/1.0
```

Figure 1

As illustrated, gaps were inserted into sequence two in order to align it to sequence one. The algorithm that is used to perform the alignment is the Needleman Wunsch algorithm. The Needleman Wunsch algorithm [3] is a global sequence alignment algorithm created in 1970.

Global Sequence Alignment

Global alignment is the act of aligning to sequences in which the two sequences are aligned from beginning to end, as opposed to local alignment algorithms that would find the strongest subsequence alignment only. Global sequence alignment is used when two sequences are already considered similar, due to the fact that if they were not, an excess number of gaps would be inserted and the alignment would lose its meaning. In the context of protocol analysis, Needleman Wunsch is used after already determining the overall similarity score between two sequences. In the example found in figure 1, one can discern that in the context of protocol analysis, the value "index.html" is a variable length field in the packet format, while "GET /" and " HTTP/1.0" are required, unchanging fields.

The Needleman Wunsch algorithm is a dynamic programming algorithm that operates on a matrix. Sequence two is placed in the top of the matrix and sequence one runs down the left side. There are three steps to the Needleman Wunsch algorithm: similarity scoring, summing and back-tracing. The algorithm

		G	E	T		/	i	n	d	e	x	.	h	t	m	l		H	T	T	P	/	1	.	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
T	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
/	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6
H	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	7	7	7	7	7	7	7	7
T	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	7	8	8	8	8	8	8	8
T	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	7	8	9	9	9	9	9	9
P	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	7	8	9	A	A	A	A	A
/	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	7	8	9	A	B	B	B	B
1	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	6	7	8	9	A	B	C	C	C
.	0	1	2	3	4	5	5	5	5	5	5	6	6	6	6	6	6	7	8	9	A	B	C	D	D
0	0	1	2	3	4	5	5	5	5	5	5	6	6	6	6	6	6	7	8	9	A	B	C	D	E

Table 1: A completed matrix processed by the Needleman Wunsch algorithm. Highlighted is the path that is traced back to the origin that is used to determine the presence of gaps in either sequence.

operates on a matrix that is N+1xM+1 in size, where N is the length of sequence one and M is equal to the length of sequence two. The matrix is initialized with zero in each cell. For the first step of performing similarity scoring, each cell in the matrix is scored based on the matching similarity between each character in sequence one and two. A typical value used to score a match is 1. Mismatches can also be scored, however for a typical Needleman Wunsch configuration, the mismatch score is usually 0. Similarity matrices, which will be described in section 3, are also used to aid in the process of calculating match scores and improving overall alignment. The second step of summing the matrix starts in cell 1,1 and each cell is evaluated using the equation found in Eq.(1).

$$M_{ij} = \max\begin{cases} M_{i-1,j-1} + S_{ij} \\ M_{i,j-1} + w \\ M_{i-1,j} + w \end{cases}$$

Eq.(1): where S is the score computed in step one and w is equal to the gap penalty.

A gap penalty is not required for the operation of the Needleman Wunsch algorithm, but is used sometimes to improve alignments between more distance sequences. For protocol analysis, a gap penalty of 0 has yielded accurate results.

The last operation is the back-trace step in which starting in the cell with the highest score, a path is followed that maximizes the alignment score back to the origin. The actual rule for back-trace, is to assess the left, upper and diagonal cell and move to the one with the maximum score. If all cells are equal we move to the diagonal cell. This concept is shown in Figure 2, starting in the cell with the highest score (0xE), the path is traced back to the origin. To actually perform the alignment, the completed matrix is read as it performs the back-trace step. The rules for alignment are as follows:

1. If moving diagonal, do nothing
2. If moving left, insert gap into sequence two
3. If moving up, insert gap into sequence one

Applying these rules to Table 1 we get the results shown in Figure 1. The actual alignment score for this particular example is 15. This value signifies the overall similarity between these two sequences. As stated previously, global alignment is used only when comparing two similar sequences. In order to first obtain a set of highly similar sequences from a packet database, a local alignment algorithm must be used.

Local Alignment

Local sequence alignment algorithms are used when comparing two highly diverged sequences. These algorithms seek to find the most similar subsequence between two sequences, signifying the shared genetic information that has been preserved through multiple evolutionary paths. The ways these algorithms are typically used are to first search for highly similar sequences that would be later analyzed using a global alignment algorithm. The same principle applies for network packets. Many network protocols have multiple message types that are used throughout the duration of a client-server session. An example of this is the Server Message Block (SMB)/NetBIOS protocol [4], in which there are more than a dozen message types. Obviously, one would not want to align two packets with different message types, as the overall alignment would most likely be meaningless. Instead, a local alignment algorithm is used to first populate a set of similar sequences for later analysis with the Needleman Wunsch algorithm.

The natural choice for a local alignment algorithm is the Smith Waterman algorithm [5]. This local alignment algorithm is derived from the Needleman Wunsch algorithm with some modifications. Unlike Needleman Wunsch, Smith Waterman requires a gap penalty to work correctly. A common Smith Waterman configuration is a match score of 2, mismatch score of -1 and a gap penalty of -2. When the initial matrix is initialized for Smith Waterman, the left most row and up most column are filled with values starting at 0 ending at 0 minus the length of the sequences. The algorithm behaves just like Needleman Wunsch except for the

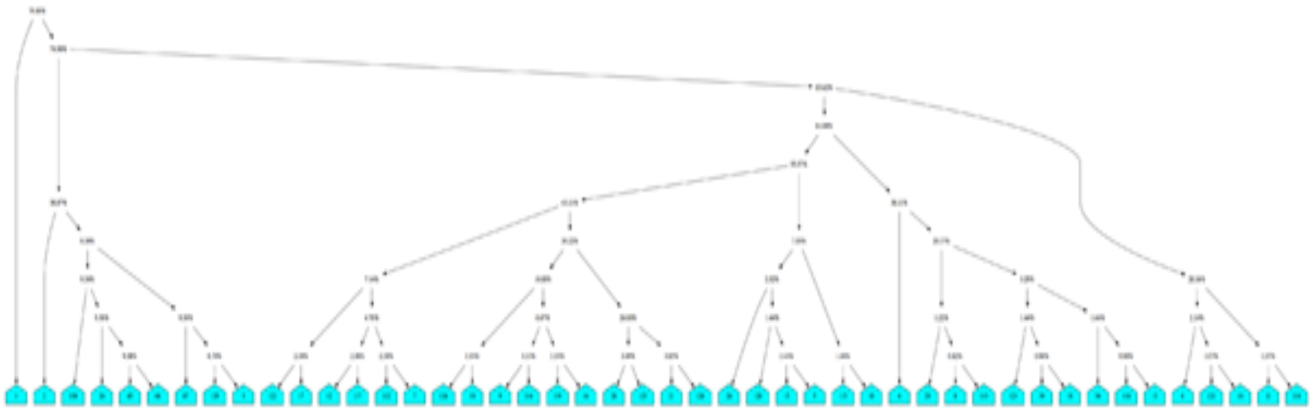

Figure 2: Phylogenetic Tree of SMB Protocol

fact that it returns from the trace-back step when it reaches a cell with a value of 0.

While local sequence alignment is not used to understand protocol fields, it allows one to cluster together a set of similar sequences for later analysis in which protocol fields will be revealed with global alignment algorithms.

Similarity Matrices

A similarity matrix is used in bioinformatics to optimize the alignment between sequences. These matrices are used during the first step of the Needleman Wunsch algorithm in which instead of scoring 1 for a match, it would perform a lookup of the similarity between two characters and place it into the appropriate cell. The most common matrices found in bioinformatics are the Percent Accepted Mutation (PAM) [8] and Blocks Substitution Matrix (BLOSUM) [2] matrices. These matrices were derived based on observation of evolutionary data on many sequences using Markov chains among other techniques. The analogous usage for protocol informatics would be to better align sequences based on actual datatypes. In network protocols, data is usually separated into plaintext data and binary data. Many protocols contain both types of data in the same packet. By creating similarity matrices based on observation of multiple popular protocols, it is possible to optimize the alignments. The general rule for matrix creation would be that binary data is more similar versus characters that fall within the ASCII set and vice versa. Other protocol characteristics could be represented by a similarity matrix such as a sequence number which would have a stronger bond with its succeeding value. By using these matrices, gaps would be inserted in the most optimal locations and allowing a better understanding of where variable length fields exist in the overall packet format.

3. MULTIPLE SEQUENCE ALIGNMENT

To better understand the structure of sequences, it is useful to align them against multiple sequences. Multiple sequence alignment algorithms are difficult in that they are not computationally feasible to really use. One could use Needleman Wunsch in which a hypercube was traversed. This however is O(N^n) therefore diminishing its usefulness. This forces biologists to adopt heuristic methods for performing multiple sequence alignment.

To perform a multiple sequence alignment, it is helpful to create a phylogenetic tree to guide the process. A phylogenetic tree is an evolutionary tree that illustrates mutations as time goes on. This is analogous to network packets in that packets mimic evolution by changing the values contained in fields.

A common method used to generate phylogenetic trees is the Unweighted Pairwise Mean by Arithmetic Averages (UPGMA) [9] algorithm. The algorithm is defined as:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} D_{pq}$$

Eq.(2): Where dij is the distance between clusters Ci and Cj and Dpq is the Smith Waterman score.

To build the tree, each sequence is placed into an individual cluster and each cluster is inserted into the universal set. The UPGMA algorithm is then used to calculate the distance between each cluster, finding two clusters where the distance is minimal. A new cluster is created by performing a union on the two most similar clusters. A new node k with child nodes i and j is then created. Lastly, the new cluster is inserted into the universal set and the two similar clusters are removed. A phylogenetic tree built using UPGMA is illustrated in figure 2.

Now that a phylogenetic tree is constructed, the tree can be used as a guide during heuristic multiple alignment. The concept for the multiple alignment is called progressive alignment. To perform progressive alignment, the most similar sequences are aligned to each other and then merged with the results as it moves towards the root of the tree, propagating new gaps down the tree as necessary. The tree traversal rules are as follows:

1. If root is null, go left and go right
2. If left contains a sequence and right contains a sequence, perform alignment and store result with least amount of gaps in root.
3. Keep track of edits on the edges of the tree.

```
1 GET /cgi-bin/whois.pl HTTP/1.0 Host: arin.net User-Agent: Opera Accept: text/xml
2 GET /index.html HTTP/1.0 Host: www.yahoo.com User-Agent: Mozilla/5.0 Accept: text/xml
3 GET / HTTP/1.0 Host: www.google.com User-Agent: IE4.0 Accept: text/xml

1 GET /cgi-bin__/whois.pl HTTP/1.0 Host: _____a__rin.net User-Agent: __Opera____ Accept: text/xml
2 GET /__i___ndex.h___tml HTTP/1.0 Host: www.yahoo__.com User-Agent: Mozilla/5.0 Accept: text/xml
3 GET /__________________ HTTP/1.0 Host: www._google.com User-Agent: ______IE4.0 Accept: text/xml

Consensus:
  GET /?????????????????? HTTP/1.0 Host: ??????????????? User-Agent: ??????????? Accept: text/xml
```

Figure 3: Before and after multiple sequence alignment including the consensus sequence.

To compute the progressive multiple alignment the following operations will occur:

Sequence One Aligned = E(1,2) + E(1', 3)
Sequence Two Aligned = E(2,1) + E(1', 3)
Sequence Three Aligned = E(3, 1')

5. APPLYING THE ALGORITHM

Now that it has been explained how to construct a multiple alignment of sequences, we can apply these concepts to two network protocols in order to understand their overall structure. For this example, the plaintext HTTP protocol will be used.

HTTP is the hypertext transport protocol used for serving web content. It is a plaintext protocol, but is not limited to transmitting plaintext only data. For the sake of length, we assume that we have captured only three samples of the protocol. The samples and results of the alignment are found in figure 3.

To analyze the results of the alignment, one could rely solely on the consensus sequence. Consensus sequences simply separate static data from dynamic data, underlying the similarities shared by all sequences. A popular way to display a consensus sequence is with gaps and without gaps. Erroneous gap insertion can cause problems when analyzing consensus sequences, therefore viewing a consensus without gaps can be beneficial.

As one may notice, information is lost using consensus sequences alone, as only the most prevalent character is represented. Dr. Tom Schneider created Sequence Logos [1] to combat this problem using principles from Shannon's information theory. Further research is being completed in order to apply sequence logos to protocol analysis.

Using the information gathered by the consensus sequence allows one to put together a rough description of the protocol. For computer security purposes, this information can be used to "fuzz" [10] the application to find security vulnerabilities. For the Protocol Informatics prototype, protocols are defined in an XML format. The XML protocol definition of HTTP based on the consensus sequence is found in figure 4.

```
<ProtocolSpec name="HTTP" transport="TCP" port="80">

<Packet name="general">
<String>GET /</String>
<Variable type="string"/>
<String> HTTP/1.1\r\nHost: </String>
<Variable type="string"/>
<String> User-Agent: </String>
<Variable type="string"/>
<String> Accept: text/xml</String>
</Packet>
</ProtocolSpec>
```

Figure 4: XML protocol definition derived from consensus sequence

6. CONCLUSIONS AND FUTURE WORK

This paper detailed a method for applying bioinformatics algorithms to network protocol analysis for the purpose of identifying packet fields. While the stage in this research is early, the results thus far have been encouraging given the success of string based and binary based protocol analysis using these methods.

The next step would be to apply the sequence logo technology to take the place of the consensus sequence as information deserves to be preserved. Also, a set of similarity matrices are to be developed to aid in the analysis of plaintext, binary and mixed protocol types. These matrices will allow for stronger alignments and better automated detection of packet fields.

Currently, the prototype has been coded in python, allowing for an experimental framework throughout the research phase. Eventually, it will be desirable to either integrate this technology with an open source protocol analysis tool such as Ethereal, or write one from scratch.

Protocol Informatics is freely available and can be downloaded at http://www.baselineresearch.net/PI.

7. ACKNOWLEDGEMENTS

I thank Christopher Abad, Tom Schneider, Terry Gaasterland, Ken Steele and Michael Davis for their helpful suggestions and contributions.

8. REFERENCES

[1] T.D Schneider. **Consensus Sequence Zen**. Applied Bioinformatics, 3:111-119, 2002.
[2] Henikoff and Henikoff. **Amino acid substitution matrices from protein blocks**. PNAS 89:10915-10919, 1992.
[3] S. B. Needleman and C. D. Wunsch. **A general method applicable to the search for similarities in the amino acid sequence of two proteins**. Journal of Molecular Biology, 48:444-453, 1970.
[4] Internet Activities Board. **RFC 1001 - Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods**. 1987 March.

[5] Smith, T.F and Waterman, M.S. **Identification of common molecular subsequences.** Journal of Molecular Biology, 147: 195-197, 1981.

[6] Gerald Combs. **Ethereal Network Analyzer**. *www.ethereal.org*. Viewed on March 1st, 2005.

[7] LBNL Network Research Group. **Tcpdump**. *www.tcpdump.org*. Viewed on March 1st, 2005.

[8] Dayhoff, M.O and Schwartz, R.M and Orcutt, B.C. **A model of evolutionary change in proteins.** Atlas of protein sequence and structure, 5:345-358, 1978.

[9] Tateno Y, Nei M, Tajima F. **Accuracy of estimated phylogenetic trees from molecular data**. J Mol Evol. 18(6):387-404, 1982

[10] Dave Aitel. **Advantages of block based protocol analysis for security testing**. *www.immunitysec.com.* Viewed on March 2nd, 2005.