Cyber Security of the Application Layer of Mission Critical Industrial Systems

Rafał Kozik[1(✉)], Michał Choraś[1], Rafał Renk[2], and Witold Hołubowicz[1,2]

[1] Institute of Telecommunications and Computer Science,
UTP University of Science and Technology in Bydgoszcz, Bydgoszcz, Poland
rafal.kozik@utp.edu.pl
[2] Adam Mickiewicz University, UAM, Poznan, Poland
renk@amu.edu.pl

Abstract. In this paper we focus on proposing the effective methods of cyber protection of the application layer. We also discuss how this challenge is related to mission critical industrial and manufacturing systems. In this paper we propose two step HTTP request analysis method that engages request segmentation, statistical analysis of the extracted content and machine learning on the imbalanced data. In this work, we particularly addressed the segmentation technique that allows us to divide the large dataset on smaller subsets and learn the classifiers in a significantly shorter time. In our experiments we evaluated several classifiers that are popular in data mining community. The results of our experiments are obtained on a benchmark CSIC'10 HTTP dataset. The proposed approach allows us to further improve the achieved results of protecting application layer in comparison to other benchmark approaches.

Keywords: Cyber security · Anomaly detection · Pattern extraction · Application layer attacks · Web application security

1 Introduction and Rationale

The problem of cyber security in the application layer is recently more severe and challenging. In fact, top-ranked network threats and attacks e.g. on the OWASP list are those targeting application layer (such as SQLIA and XSS [2]). There are many reasons for such situation such as programmers faults, software bugs but also large number of the new applications and services being launched every day, their uniqness, lack of security standards for web applications, rapid changes in scalability etc. Another key aspect is the lack of cyber security awareness within the users and very often users are those providing access and creating security holes being the weakest link in the security chain.

On the other hand, the level of cyber security of operating systems and network protocols is constantly increasing. Therefore, for the attackers it is cheaper, easier and more effective to focus on attacks on the application layer.

K. Saeed and W. Homenda (Eds.): CISIM 2016, LNCS 9842, pp. 342–351, 2016.
DOI: 10.1007/978-3-319-45378-1_31

In this paper we focus on the effective methods of anomaly detection in the application layer. We also discuss how cyber threats and attacks in the application layer are related to mission critical industrial and manufacturing systems.

This paper is structured as follows. First, we discuss the cyber security aspects of mission critical industrial and manufacturing systems with the focus on the application layer. Later we provide an general overview of proposed method (Sect. 3). Then, the detailed description of request segmentation and feature vector encoding is given (Sect. 4). The experiments and results are given afterwards.

2 Related Work

There are many solutions (called web application firewalls - WAFs), that work as signature-based filters (e.g. set of rules) over the HTTP traffic. These signatures usually cover common web application attacks (e.g. SQL Injection, Cross Site Scripting, etc.). In example, the ModSecurity [3] plug-in for Apache web server (particularly the "OWASP ModSecurity Core Rule Set – CRS") is intended to provide defensive protection against common application layer attacks. Also the PHPIDS [4] tool is another security measure for web applications. However, in contrast to ModSecurity it can be integrated only with PHP-enabled web servers. The NAXSI [5] is a third party plug-in module for high performance Nginx [6] web server. The NAXSI stands for Nginx Anti XSS and SQL Injection and it provides low-level rules that detect keywords (symptoms) of application layer attacks. In contrast to ModSecurity and PHPIDS, NAXSI learns normal application behaviour instead of attacks patterns.

In the literature there are also methods applying anomaly detection (variety of complex schemas for learning normal/anomalous models). For instance, in [7] authors used χ^2 metric and character distribution approach to detect anomalous HTTP requests. In order to increase the attack detection effectiveness authors used a parser that splits the request into URL address and query string of attributes followed by values. Another approach to HTTP traffic anomaly detection was presented in [8]. Authors applied DFA (Deterministic Finite Automaton) to compare the requests described by the means of tokens. In contrast, in [9] authors have compared different n-grams techniques applied to application layer anomaly detection. In the literature there are also approaches combining n-grams with Self Organizing Maps [10], Bloom filters [11], and wide variety of different machine-learnt classifiers [12].

3 Can Mission Critical Industrial Systems Be Targeted by Application Layer Cyber-Attacks?

So far in computer security and critical infrastructures protection community, the focus was on cyber protection of network protocols in the lower layers. Also, after

the successful attack on Ukrainian power system in December 2015, the severity of such attacks is well known. Therefore, many researchers and projects were focused on SCADA/ICS cyber protection [30–32], WSN protection etc. [33,34].

However, mission critical cyber physical systems and critical infrastructures can be also targeted by cyber attacks on the application layer and web services/applications.

The cyber attacks targeting web applications are usually cross domain, meaning that they are targeting information systems such as web servers or web services, but also indirectly can impact physical infrastructures (e.g. application controlling industrial and manufacturing processes). This happens due to the increasing interconnectivity among different systems, but also due to the fact that more and more institutions are adapting internet technologies to provide new services on top of the existing infrastructure. In many cases, the legacy systems were never expected nor designed to be connected via open network.

One example of the problems with complex interconnectivity is the Havex worm case [1]. It used the security hole in web application of company providing software solutions for SCADA systems. The attacker was able to infect genuine software installers that were used to manage the SCADA systems. The infection was part of so called "water drop" attack, where the attacker wants to remain hidden as long as possible, infecting only the interested party, in this case electrical plants.

Another examples of the successful attacks on critical infrastructures (usually databases and information systems) are attacks on Italian [27] and Turkish [28] governmental databases or alleged attack that paralysed LOT Polish Airlines [29] computer system resulting even in cancelled flights.

One of the main challenges related to the web-layer attacks is fact that those are difficult to detect with typical signature-based approaches. Therefore, there is a need for methods that will use the different approach. In this paper, we propose an anomaly-based solution that analyses the content of the HTTP payload. Upon that, data algorithm builds an model of client web browser behaviour in order to recognize possible trials of the attack.

4 The Proposed Approach

The major contribution and innovation of this paper is the segmentation approach for anomaly detection in the application layer. The proposed approach is designed to work with the typical HTTP-based, request-response web application. In the current implementation it works as additional software supporting web server administrator. It is an anomaly detection tool that receives HTTP requests, analyses their content and classifies it either as normal or anomalous. Moreover, in this approach, we do not intend to block any traffic coming from user browser to web server. Such reaction is out of scope of our work and would need to be realized in accordance to legal requirements and service level agreements.

The general overview of information flow is presented in Fig. 1.

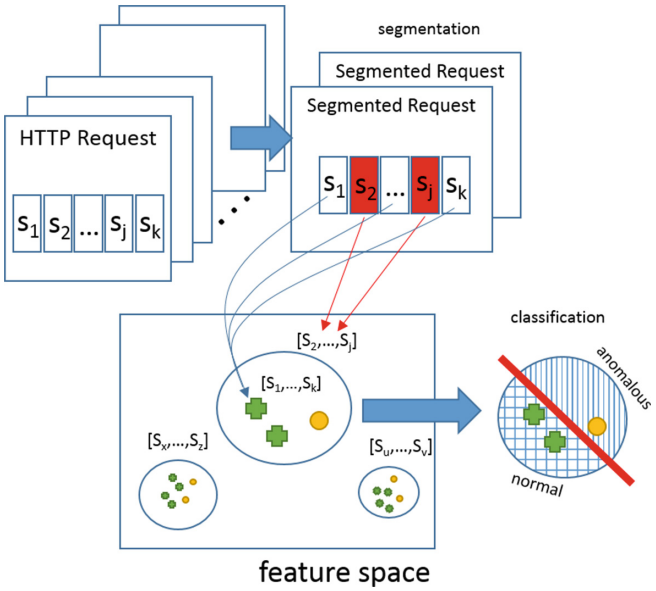

Fig. 1. The proposed two-step segmentation schema overview. The first step is a textual segmentation of similar requests. The second step is a segmentation on normal/anomalous binary class achieved with the machine-learnt classifier.

Using the available set of HTTP requests, we apply two step segmentation. The first step splits the large dataset into smaller pieces, so that further computations are executed in parallel. The segmentation uses a request segmentation method described in Sect. 4.1. It allows us to identify the possible structure of the analysed requests. The structure is described using the sequence of tokens.

The tokens (shaded boxes in Fig. 1) are used to identify clusters in our feature space that is composed of the feature vectors extracted from the data that had been tokenized (delimited by tokens). When each feature vector is extracted and assigned to the appropriate cluster, we use machine-learnt classifier to assign it to normal or anomalous class. In this research we assume that fully labelled learning dataset is available. Therefore, we can use the supervised methods.

4.1 Request Segmentation

In our work, we have proposed the technique for HTTP request segmentation. It engages dictionary of words to identify tokens (common character sequences) that appear in all the analysed request sequences. In order to find such a collection of tokens we have adapted LZW compression method (Lempel-Ziv-Welch [15,16]) that allows us to calculate the dictionary D containing characters sequences that appear in HTTP requests. The procedure follows the Algorithm 1.

This algorithm accepts a S set of HTTP payloads in order to produce a dictionary D, which will contain list of words (character sequences). The

dictionary has a form of unordered list. Each position on that list can be taken only by one word. Obviously, using the dictionary D, one can compress the data replacing words (characters sequences) with a numbers that will correspond to the positions of that sequences in the dictionary. For efficiency reasons the D is implemented as a hash-table to provide $O(1)$ lookup time. The algorithm for establishing dictionary iteratively scans the set S of payloads until all the data is processed. In each iteration a lookup string s is built. At the beginning, it is an "empty string" that is further extended (in each step) with a single character ch read from S set. However, the s lookup string is extended with ch only if "$s+ch$" is found in the dictionary. If it is not found, the dictionary D is updated with new word "$s + ch$", s is reduced to ch and the procedure starts over.

Data: Set of HTTP payloads S
Result: Dictionary D
s = empty string
while *there is still data to be read in* S **do**
  $ch \leftarrow$ read a character ;
  if $(s + ch) \in D$ **then**
    $s \leftarrow s+ch$;
  else
    $D \leftarrow D \cup (s + ch)$;
    $s \leftarrow ch$;
  end
end

Algorithm 1. Algorithm for establishing dictionary D.

In our experiments we only analyse the HTTP traffic. Therefore, we can leverage two mandatory elements of the request, namely HTTP request method (GET, POST, PUT, UPDATE, DELETE are the most frequently used ones) and the URL address of the resources the method is executed. In result, we can build the dictionary separately per different HTTP methods and URL addresses.

Additionally, in order to decrease the dimensionality of the feature vectors in a given component we post-process the dictionary eliminating (removing) those entries that are sub-sequences of other sequences. In other words, we prefer longer tokens over shorter. Once the dictionaries are established, we identify their positions in analysed requests. To correctly extract feature vector from a request, we firstly must identify the correct sequence of tokens that appears in the whole group of the analysed requests. To achieve that we use progressive multiple sequences alignment algorithm [14]. Finally, we build the feature vectors for each analysed HTTP request. However, we are using only those characters that do not belong to any token.

4.2 Parameters Encoding

In this work we have used typical approach for textual data encoding that adapts character distribution histograms (see Fig. 2). However, instead of classical byte

per bin association, we count the number of characters such that the decimal value in an ASCII table belongs to following ranges: $<0, 31>$, $<32, 47>$, $<48, 57>$, $<58, 64>$, $<65, 90>$, $<91, 96>$, $<97, 122>$, $<123, 127>$, $<128, 255>$. Selected ranges represent different type of symbols like numbers, quotes, letters or special characters (see Table 1). In result our histogram will have 9 bins. This is significant dimensionality reduction in contrast to sparse 256-bin histograms (one bin per byte character).

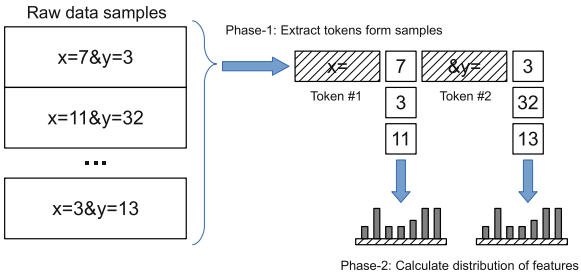

Fig. 2. The high-level overview of parameter encoding approach.

Table 1. ASCII code ranges and corresponding characters.

Group	Range	Characters
1	$< 0, 31 >$	`control characters`
2	$< 32, 47 >$	`SPACE!"#$%&'()*+,-./`
3	$< 48, 57 >$	`0123456789`
4	$< 58, 64 >$	`:;<=>?@`
5	$< 65, 90 >$	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
6	$< 91, 96 >$	`[\]^_‘`
7	$< 97, 122 >$	`abcdefghijklmnopqrstuvwxyz`
8	$< 123, 127 >$	`{~}|`
9	$< 128, 255 >$	`special characters`

5 Experiments and Results

For the experiments, the CSIC'10 dataset [18] was used. It contains several thousands of HTTP protocol requests which are organised in the form similar to the Apache Access Log. The dataset was developed at the Information Security Institute of CSIC (Spanish Research National Council) and it contains the generated traffic targeted to an e-Commerce web application. For convenience, the data was split into anomalous, training, and normal sets. There are over

Table 2. Effectiveness of the proposed approach evaluated for different classifiers

Method	TP rate	FP rate	Precision	F-measure
RandomForest	0.996	0.061	0.85	0.917
NaiveBayes	0.944	0.037	0.899	0.921
AdaBoost	0.935	0.002	0.994	0.964
PART	0.984	0.03	0.917	0.949
J48	0.991	0.05	0.873	0.928
Our previous work	0.977	0.081	0.809	0.885
Nguyen (avg.) [19]	0.936	0.069	-	-

36000 normal and 25000 anomalous requests. The anomalous requests refer to a wide range of application layer attacks, such as: SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS, server side include, and parameter tampering.

Moreover, the requests targeting hidden (or unavailable) resources are also considered as anomalies. Some examples classified to this group of anomalies include client requests for: configuration files, default files or session ID in URL (symptoms of http session taking over attempt). What is more, the requests, which parameters do not have appropriate format (e.g. telephone number composed of letters), are also considered anomalous. As authors of the dataset explained, such requests may not have a malicious intention, but they do not follow the normal behaviour of the web application.

According to authors knowledge, there is no other publicly available dataset for web attack detection problem. The datasets like DARPA or KDD'99 are outdated and do not include many of the actual attacks.

The results presented in Table 2 contain measures of the following characteristics:

- *TP Rate* - True Positive Rate indicating the percentage of detected HTTP request labelled as anomalous.
- *FP Rate* - False Positive Rate indicating the percentage normal HTTP request labelled (wrongly) as anomalous.
- *Precision* - number of True Positives divided by sum of False Positives and True Positives.
- *F-Measure* - calculated according to following equation:

$$F - Measure = 2 \cdot \frac{Precision * Recall}{Precision + Recall} \tag{1}$$

In this equation *Recall* is equivalent to *TP Rate*.

The first five rows in the Table 2 refer to the proposed method, while the remaining two serve as a baseline for the measurements, namely our older previous approach [17] and the results reported by CSIC'10 dataset authors in [19].

For the proposed method we evaluated different classifiers. Herby we used RandomForest, NaiveBayes, AdaBoosting (with decision stumps as weak classifiers), PART, and J48 classifiers.

As it can be noticed, the proposed method allows us to achieve better precision in all cases. In case of [19] the values have not been provided by authors. In case of a PART and J48 tree classifier, we were able to improve the TPR and FPR values in comparison to our previous work and to results reported in [19].

6 Conclusions

In this paper we have presented the improved method of detecting attacks and anomalies in the application layer. Herby, we have proposed a two-step HTTP request segmentation. First, we apply a textual analysis of the request in order to identify the possible structure. Then, we apply feature extraction method and machine-learnt classifier.

In our work we focused on HTTP request segmentation in order to obtain better effectiveness and scalability. Conducted experiments show that this approach allows us to achieve higher recognition rates of attacks while having lower rates of false positives. Moreover, the segmentation step before the classification allows us to divide dataset on smaller ones, thus decreasing the time required for learning the classifier.

The reported results have been achieved for benchmark CSIC'10 HTTP dataset. Additionally, we have compared our results with those reported by authors of the dataset. As it was presented, our modification allows us to achieve better results.

Acknowledgments. The CIPRNet project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 312450. The European Commission's support is gratefully acknowledged.

The work is also funded by the Polish National Centre for Research and Development (NCBiR) from funds for science in the years 2013–2016, allocated for the international projects.

References

1. F-Secure. Backdoor: W32/HAVEX description. https://www.f-secure.com/v-descs/backdoor_w32_havex.shtml
2. OWASP Top. 10 2013. OWASP project homepage. https://www.owasp.org/index.php/Top_10_2013-Top_10
3. ModSecurity project homepage. https://www.modsecurity.org/
4. PHPIDS project homepage. https://github.com/PHPIDS/PHPIDS
5. NAXSI project homepage. https://github.com/nbs-system/naxsi
6. NGINX project homepage. http://nginx.org/en/

7. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of the 10th ACM conference on Computer and communications security, pp. 251–261 (2003)
8. Ingham, K.L., Somayaji, A., Burge, J., Forrest, S.: Learning DFA representations of HTTP for protecting web applications. Comput. Netw. **51**(5), 1239–1255 (2007)
9. Hadžiosmanović, D., Simionato, L., Bolzoni, D., Zambon, E., Etalle, S.: N-Gram against the Machine: On the Feasibility of the N-Gram network analysis for binary protocols. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 354–373. Springer, Heidelberg (2012)
10. Bolzoni, D., Zambon, E., Etalle, S., Hartel, P.H.: POSEIDON: a 2-tier anomaly-based Network Intrusion Detection System. In: IWIA 2006: Proceedings of 4th IEEE International Workshop on Information Assurance, pp. 144–7156 (2006)
11. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: a content anomaly detector resistant to mimicry attack. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 226–248. Springer, Heidelberg (2006)
12. Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., Lee, W.: McPAD: a multiple classifier system for accurate payload-based anomaly detection. Comput. Netw. **53**(6), 864–881 (2009)
13. Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. ACM SIGCOMM Comput. Commun. Rev. **34**, 357–374 (2004)
14. Higgins, D.G., Sharp, P.M.: CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene **73**(1), 237–244 (1988)
15. Welch, T.: A technique for high-performance data compression. IEEE Comput. **17**(69), 8–19 (1984)
16. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Trans. Inf. Theory **23**, 337–343 (1977)
17. Kozik, R., Choras, M., Renk, R., Holubowicz, W.: Patterns extraction method for anomaly detection in HTTP traffic. In: Herrero, A., Baruque, B., Sedano, J., Quintan, H., Corchado, E. (eds.) CISIS 2015 and ICEUTE 2015. AISC, vol. 369, pp. 227–236. Springer, Heidelberg (2015)
18. Torrano-Gimnez, C., Prez-Villegas, A. lvarez G.: The HTTP dataset CSIC 2010 (2010). http://users.aber.ac.uk/pds7/csic_dataset/csic2010http.html
19. Nguyen, H.T., Torrano-Gimenez, C., Alvarez, G., Petrović, S., Franke, K.: Application of the generic feature selection measure in detection of web attacks. In: Herrero, Á., Corchado, E. (eds.) CISIS 2011. LNCS, vol. 6694, pp. 25–32. Springer, Heidelberg (2011)
20. Sharma, M., Toshniwal, D.: Pre-clustering algorithm for anomaly detection and clustering that uses variable size buckets. In: 2012 1st International Conference on Recent Advances in Information Technology (RAIT), pp. 515–519, 15–17 March 2012
21. Adaniya, M.H.A.C., Lima, M.F., Rodrigues, J.J.P.C., Abrao, T., Proenca, M.L.: Anomaly detection using DSNS and fireflyharmonic clustering algorithm. In: 2012 IEEE International Conference on Communications (ICC), pp. 1183–1187, 10–15 June 2012
22. Mazel, J., Casas, P., Labit, Y., Owezarski, P.: Sub-space clustering, inter-clustering results association and anomaly correlation for unsupervised network anomaly detection. In: 2011 7th International Conference on Network and Service Management (CNSM), pp. 1–8, 24–28 October 2011

23. Yang, C., FeiqiDeng, H.Y.: An unsupervised anomaly detection approach using subtractive clustering and hidden markov model. In: Second International Conference on Communications and Networking in China, CHINACOM 2007, pp. 313–316, 22–24 August 2007
24. Liang, H., Wei-wu, R., Fei, R.: An adaptive anomaly detection based on hierarchical clustering. In: 2009 1st International Conference on Information Science and Engineering (ICISE), pp. 1626–1629, 26–28 December 2009
25. Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. Graph Algorithms Appl. **10**(2), 191–218 (2006)
26. Liao, Q., Blaich, A., Van Bruggen, D., Striegel, A.: Managing networks through context: graph visualization and exploration. Comput. Netw. **54**, 2809–2824 (2010)
27. Cyberattack on Italian government. http://www.lastampa.it/2015/05/19/italia/cronache/anonymous-colpisce-il-ministero-della-difesa-qlFNgswyvu20wnQiNYK1kL/pagina.html
28. Cyberattack on Turkish government. http://www.ehackingnews.com/2013/06/istanbul-special-provincial.html
29. Cyberattack on Polish Airlines LOT company. http://uk.reuters.com/article/2015/06/21/uk-poland-lot-cybercrime-idUKKBN0P10WY20150621
30. Coppolino, L., et al.: Enhancing SIEM technology to protect critical infrastructures. In: Hämmerli, B.M., Svendsen, N.K., Lopez, J. (eds.) Critical Information Infrastructures Security, vol. 7722, pp. 10–21. Springer, Heidelberg (2013)
31. Collins, S., McCombie, S.: Stuxnet: the emergence of a new cyber weapon and its implications. J. Policing Intell. Counter Terrorism **7**(1), 80–91 (2012)
32. Takagi, H., et al.: Strategic security protection for industrial control systems. In: 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). IEEE (2015)
33. Romano, L., D'Antonio, S., Formicola, V., Coppolino, L.: Protecting the WSN zones of a critical infrastructure via enhanced SIEM technology. In: Ortmeier, F., Daniel, P. (eds.) SAFECOMP Workshops 2012. LNCS, vol. 7613, pp. 222–234. Springer, Heidelberg (2012)
34. Formicola, V., et al.: Assessing the impact of cyber attacks on wireless sensor nodes that monitor interdependent physical systems. In: Butts, J., Shenoi, S. (eds.) Critical Infrastructure Protection VIII, vol. 441, pp. 213–229. Springer, Heidelberg (2014)