

ASAP: Automatic Semantics-Aware Analysis of Network Payloads

Tammo Krueger^{1,3}, Nicole Krämer^{2,3}, and Konrad Rieck³

¹ Fraunhofer Institute FIRST, Germany

² Weierstrass Institute for Applied Analysis and Stochastics, Germany

³ Berlin Institute of Technology, Germany

Abstract. Automatic inspection of network payloads is a prerequisite for effective analysis of network communication. Security research has largely focused on network analysis using protocol specifications, for example for intrusion detection, fuzz testing and forensic analysis. The specification of a protocol alone, however, is often not sufficient for accurate analysis of communication, as it fails to reflect individual semantics of network applications. We propose a framework for semantics-aware analysis of network payloads which automatically extracts semantics-aware components from recorded network traffic. Our method proceeds by mapping network payloads to a vector space and identifying communication templates corresponding to base directions in the vector space. We demonstrate the efficacy of semantics-aware analysis in different security applications: automatic discovery of patterns in honeypot data, analysis of malware communication and network intrusion detection.

1 Introduction

Automatic analysis of network data is a crucial task in many applications of computer security. For example, intrusion detection systems often require parsing of network payloads for identification of attacks [1–3], fuzz testing tools build on automatically crafting network messages from protocol specifications [4–6], and forensic analysis depends on inspecting network data involved in security incidents [7–9]. In these and several other security applications, the analysis of communication—whether from live traffic or recorded traces—critically depends on automatic extraction of meaningful patterns from network payloads, such as application parameters, cookie values and user credentials.

A large body of security research has thus focused on analysis of network data using protocol specifications [10–12]. The specification of a protocol defines the basic structure of its communication as well as the syntax of its network messages. While these analysis techniques are successful in parsing network data, they are by design confined to the examination of protocol syntax. However, attacks and security threats are rarely reflected in syntax alone but in semantics, functionality realized on top of protocol specifications. As an example, malicious software often employs standard network protocols for communication. However, parsing of corresponding network traffic is not sufficient for accurate analysis,

and significant manual effort is necessary for deducing relevant information from parsed content. What is needed are techniques capable of automatically identifying and extracting semantics-aware components from communication, thereby reducing the gap between protocol syntax and semantics.

In this paper, we propose a framework for automatic, semantics-aware analysis of network payloads (ASAP). This framework is orthogonal in design to specification-based approaches and automatically extracts semantics-aware components from recorded traffic—even if the underlying protocols are unknown. To this end, ASAP ignores any protocol specification and exploits occurrences and combinations of strings for inferring communication templates which are more focused on the semantics instead of the syntax of communication. The main contributions of this framework are:

1. *Alphabet extraction for network payloads.* For automatic analysis, we devise a technique for extracting an alphabet of strings from network payloads. The alphabet concisely characterizes the network traffic by filtering out unnecessary protocol or volatile information via a multiple testing procedure and embeds the payloads into a vector space.
2. *Analysis using matrix factorization.* The method proceeds by identifying base directions in this vector space using concepts of matrix factorization, which combines these letters to meaningful building blocks.
3. *Construction of communication templates.* Discovered base directions are transformed to communication templates—conjunctions of strings—which give insights into semantics of communication and provide a basis for interpretation of traffic beyond syntax-based analysis.

Empirically, we demonstrate the capabilities of semantics-aware analysis in different security applications. First, we conduct experiments on network traffic captured using honeypots where we pinpoint exploited vulnerabilities as well as attack sources using the ASAP framework. Second, we apply semantics-aware analysis for investigation of network traces recorded from malware executed in sandbox environments. We exemplarily extract typical communication patterns for the malware *Vanbot*. Finally, we employ our framework in the domain of network anomaly detection by mapping payloads into a low-dimensional space without accuracy loss yet significantly increased runtime performance.

2 The ASAP Framework

The ASAP framework proceeds in three analysis stages, which are outlined in Figure 1. First, an alphabet of relevant strings is extracted from raw network payloads and used to map these payloads into a vector space for analysis. Second, concepts of matrix factorization are applied for identification of base directions in the vector space, characterizing usage patterns of mapped payloads. Third, each of these base directions is traced back to a conjunction of strings from the underlying alphabet and results in a template of typical communication content.

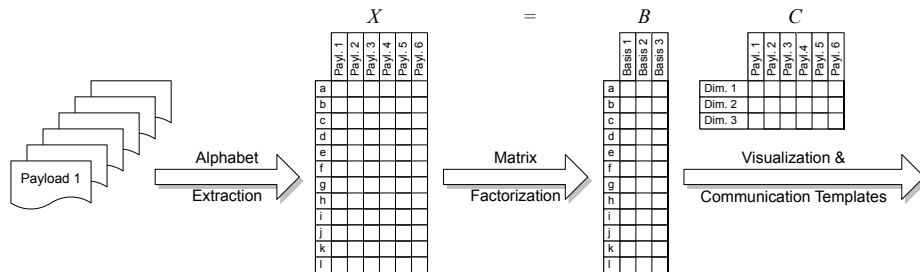


Fig. 1: Overview of the ASAP framework.

2.1 Alphabet Extraction for Network Payloads

A payload p is a string of bytes contained in network communication. For describing and characterizing the content of a payload, we automatically extract an *alphabet* S of relevant strings from a set of payloads which provides the ground for inferring communication templates. The alphabet S is initially constructed from a set of basic strings and then refined using filtering and correlation techniques.

Basic strings. Depending on the network data to be analyzed, we build the alphabet from a different set of basic strings. If we consider a protocol with distinct delimiter bytes, such as HTTP, SMTP or FTP, we base the alphabet on *tokens*—the set of all strings separated by delimiters. For a set of delimiter bytes D , such as space or carriage return, tokens can be defined as

$$S = \{\{0, \dots, 255\} \setminus D\}^*.$$

However, for binary network protocols such as DNS, SMB and NFS, we need to define the basic strings differently, as no delimiter symbols are available. In these cases we apply the concept of *n-grams* which denotes the set of all strings with fixed length n . Formally, this set of basic strings can be defined as

$$S = \{0, \dots, 255\}^n.$$

Alphabet filtering and correlation. Strings within network payloads naturally appear with different frequency, ranging from volatile to constant occurrences. For instance, every HTTP request is required to contain the string HTTP in its header, whereas other parts such as timestamps or session numbers are highly variable. Both, constant and highly volatile components, do not augment semantics and thus are filtered from the alphabet S . More precisely, we employ a statistical *t-test* for identifying non-constant and non-volatile strings by testing whether their frequency is significantly different from 0 and 1. We apply the correction proposed by Holm [13] to avoid problems with multiple testing.

With the remaining alphabet, we apply a correlation analysis to combine co-occurring strings. That is, for each string $s \in S$ we compute the Pearson correlation coefficient to all other strings and group strings which are highly

correlated, i.e. have a correlation coefficient of roughly one. Hence, we combine elements of S which co-occur in the analyzed data and thereby further refine our alphabet. Together, filtering and correlation compact the alphabet and lead to a focused representation of S ignoring both static and volatile information.

Map to vector space. Using the alphabet S , we map a network payload p to an $|S|$ -dimensional vector space, such that each dimension is associated with a string $s \in S$. In particular, we define a mapping function ϕ as

$$\phi : P \rightarrow \{0, 1\}^{|S|}, \quad \phi : p \mapsto (I(s, p))_{s \in S},$$

where P is the domain of all considered payloads and $I(s, p)$ an indicator function returning 1 if the string s is contained in p and 0 otherwise. Note, that the mapping ϕ is *sparse*, that is, the vast majority of dimensions is zero, allowing for linear-time algorithms for extraction and comparison of vectors [14].

2.2 Matrix Factorization

The mapping of network payloads to a vector space induces a geometry, reflecting characteristics captured by the alphabet S . For instance, payloads sharing several substrings appear close to each other, whereas network payloads with different content exhibit larger geometric distances. The focused alphabet of the ASAP framework enables us now to identify semantics-aware components *geometrically*. In particular, we apply the concept of matrix factorization for identifying base directions in the vector space. Given a set of payloads $P = \{p_1, \dots, p_N\}$ we first define a data matrix X containing the vectors of P as columns by

$$X := [\phi(p_1), \dots, \phi(p_N)] \in \mathbb{R}^{|S| \times N}.$$

For determining semantics-aware components, we seek a representation of X that retains most information but describes X in terms of few base directions. This can be achieved in terms of a matrix factorization of X into two matrices $B \in \mathbb{R}^{|S| \times L}$ and $C \in \mathbb{R}^{L \times N}$ such that $L \ll |S|$ and

$$X \approx BC = \overbrace{[b_1 \dots b_L]}^{\text{basis}} \underbrace{[c_1 \dots c_N]}_{\text{coordinates}}. \quad (1)$$

The columns $b_1, \dots, b_L \in \mathbb{R}^{|S|}$ of B form a new basis for the N payloads, where the dimensions of each base direction b_i are associated with the alphabet S . As we show in later experiments, this relation of base directions and the alphabet can be exploited to construct communication templates from a matrix factorization. The columns $c_1, \dots, c_N \in \mathbb{R}^L$ of C form a new set of coordinates for the payloads in a low-dimensional space, which can be used for visualization.

In general, matrix factorization methods differ in the constraints imposed on the matrices B and C . In this paper, we study two standard techniques widely

used in the field of statistics and data analysis: Principal Components Analysis (PCA) [15, 16] and Non-negative Matrix Factorization (NMF) [17].

In PCA, we seek base directions, which are orthogonal and capture as much of the variance inside the data as possible. Formally, the i th direction b_i consecutively maximizes the variance of $X^\top b_i$ under the constraint that all base directions are mutually orthonormal:

$$b_i = \arg \max_{\|b\|=1} \text{var}(X^\top b) \quad \text{s.t. } b \perp b_j, j < i.$$

In NMF, the orthogonality constraints are replaced by the requirement that the matrix B and C only contain non-negative entries. Non-negative entries in the basis vectors are a more natural representation for sequential data, as each string contributes positively to the basis representation. For a fixed dimensionality L , the factorization (1) is defined in terms of the minimization criterion

$$(B, C) = \arg \min_{B, C} \|X - BC\| \quad \text{s.t. } b_{ij} \geq 0, c_{jn} \geq 0.$$

2.3 Construction of Communication Templates

After identification of base directions B in the vector space, every payload can be expressed as a tuple of coordinates. For the interpretation, it is now crucial to find a re-mapping of these coordinates to a meaningful representation, that can be used to judge semantical content of network communication.

In case of tokens as basic strings of the alphabet we can simply select all tokens exceeding a specific threshold inside the base directions for constructing a template. For alphabets of n-grams we can try to concatenate occurring n-grams and regain parts of the original ordering. For example, if we have a basis containing the 3-grams $\text{Hos} \wedge \text{ost} \wedge \text{st}$: we can easily infer, that these tokens overlap and can be concatenated to Host :

Obviously, there is no guarantee against false concatenation. Therefore we propose a greedy algorithm, which takes the calculated values for each token of the alphabet into account: By sorting the n-grams according to their assigned weights and using this inherent ordering for the matching process, we ensure a data-driven reassembly of n-grams. We first pick the token with the highest weight and look for overlaps to the other tokens, which are also ordered by their respective weights. If we find an overlap, we merge the n-grams and remove the corresponding token from the list of pending tokens. With this merged token we restart the procedure until no more overlaps are found. This token is then added to the representation list and the procedure is repeated for the next token with the highest value left, until no more tokens are in the pending list.

3 Experiments and Applications

After presenting the ASAP framework, we turn to an empirical evaluation of its capabilities in different security applications. First, we study the framework on

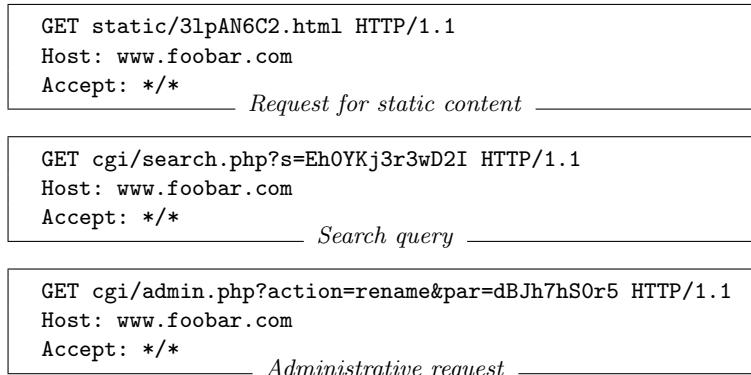


Fig. 2: Example payloads of the artificial dataset.

a toy dataset, which allows us to establish an understanding of how communication templates are inferred from communication (Section 3.1). We then proceed to real-world applications, where network traces containing malicious communication are analyzed for interesting components, such as exploited vulnerabilities and attack sources (Section 3.2 and 3.3). Finally, we apply our framework in the field of network anomaly detection by reducing the processing of network data using communication templates (Section 3.4).

3.1 A Showcase Analysis

For our first experiment, we consider an artificial dataset of HTTP communication where we have total control over protocol syntax and semantics. We simulate a web application supporting three different types of requests, whose network payloads are depicted in Figure 2. The first payload reflects a request for static content, the second payload resembles a search query and the last payload corresponds to an administrative request, in which the action parameter is one of the following `rename`, `move`, `delete` or `show`. All requests are equipped with random parts (the name of the static web page, the search string and the administration parameter) to simulate usual fluctuation of web traffic.

Using this web application, we generate a dataset of 1,000 network payloads with a uniform distribution of the three request types. We then apply the ASAP framework to this dataset as detailed in Section 2.1–2.3 using tokens as basic strings with delimiters selected according to the specification of HTTP. Based on the extracted alphabet, we then apply matrix factorization algorithms, namely Principal Component Analysis (PCA) and Non-negative Matrix Factorization (NMF) for determining base directions in the vector space of payloads. Finally, we construct communication templates for these base directions.

The extracted alphabet S consists of 8 “letters” (tokens combined by the co-occurrence analysis are grouped by brackets and the \wedge operator):

$$S = \{\text{static}, \text{cgi}, (\text{search.php} \wedge \text{s}), (\text{action} \wedge \text{admin.php} \wedge \text{par}), \text{rename}, \text{move}, \text{delete}, \text{show}\}.$$

Table 1: Templates extracted for the artificial dataset. The templates have been constructed using tokens as basic strings and NMF as factorization.

Communication Templates	
1)	static
2)	cgi \wedge (search.php \wedge s)
3)	cgi \wedge (action \wedge admin.php \wedge par)
4)	cgi \wedge (action \wedge admin.php \wedge par) \wedge move
5)	cgi \wedge (action \wedge admin.php \wedge par) \wedge rename
6)	cgi \wedge (action \wedge admin.php \wedge par) \wedge delete
7)	cgi \wedge (action \wedge admin.php \wedge par) \wedge show

Note that the alphabet does not contain tokens related to HTTP syntax or highly volatile parts of the data and thereby concentrates the following analysis on parts, which are more likely to capture the semantics of the application.

Results for the application of matrix factorization algorithms to the artificial dataset are visualized in Figure 3. For the algorithms PCA and NMF, base directions (matrix B) are shown, where the x-axis details the different directions and the y-axis the contribution of individual alphabet symbols.

While both techniques perform a matrix factorization of the payload data, the matrices differ significantly. PCA yields positive and negative contributions in the matrix B indicated by different colors. Although a certain structure and relation of alphabet symbols may be deduced from the matrix, a clear separation of different elements is not possible. By contrast, the NMF matrix shows a crisp representation of the base directions. Static and search requests are clearly reflected in individual base directions. The remaining base directions correspond to administrative requests, where different combinations of action types and other alphabet symbols have been correctly identified.

Due to this superior performance, we restrict our analysis to base directions determined using the NMF algorithm in the following. Communication templates resulting from the NMF matrix in Figure 3 are presented in Table 1. The templates accurately capture the semantics implemented in the example application. A set of 7 templates is constructed which covers static access of web content,

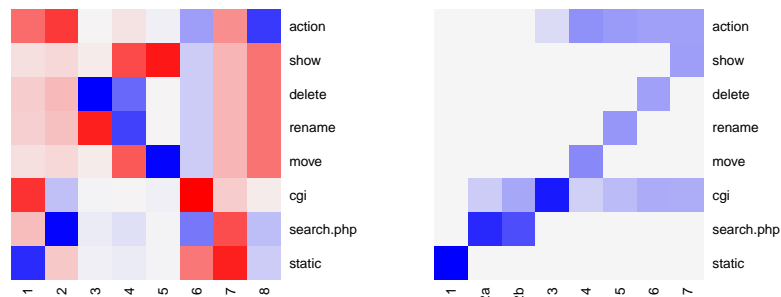


Fig. 3: Visualization of bases for PCA (left) and NMF (right) on the artificial dataset. Colors signify the intensity of the entry ranging from -1 (red) to 1 (blue).

search queries and different administrative tasks. Note that two base directions in Figure 3 are identical, resulting in a total of 7 templates. The templates even exhibit hierarchical structure: template 3 resembles a basic administrative request with all following templates being special cases for particular administrative actions, which renders NMF as prominent candidate for the construction of easily comprehensible communication templates.

3.2 Analysis of Honeypot Data

Network honeypots have proven to be useful instruments for identification and analysis of novel threats. Often however, the amount of data collected by honeypots is huge, such that manual inspection of network payloads becomes tedious and futile. The proposed ASAP framework allows for analyzing such large datasets of unknown traffic and extracts semantically interesting network features automatically.

We illustrate the utility of our framework on network data collected using the web-based honeypot *Glastopf* (<http://glastopf.org>). The honeypot captures attacks against web applications, such as remote file inclusions (RFI) and SQL injection attacks, by exposing typical patterns of vulnerable applications to search engines. The honeypot has been deployed over a period of 2 months and collected on average 3,400 requests per day. For our experiments, we randomly pick 1,000 requests from the collected data and apply our framework using tokens as underlying alphabet. In particular, we extract 40 communication templates using the base direction identified by NMF from the embedded HTTP payloads. The templates are shown in Table 2. Note that 12 templates have been omitted as they contain redundant or unspecific information.

The extracted communication templates can be classified into three categories: *semantics of malware*, *vulnerabilities* and *attack sources*. For example, the first templates reflect different options supplied to a web-based malware. Malicious functionality such as preparing a remote shell (`shellz`), setting up an IRC bouncer (`psybnc`) or scanning for vulnerable hosts (`scannerz`) are clearly manifested in strings of the templates. The following templates characterize vulnerabilities of web applications including corresponding file and parameter names. Finally, the last set of templates corresponds to domain and host names used as sources of remote file inclusions. Often not only the originating host but also parts of the complete URL have been discovered.

Note that although the communication templates have been generated from raw HTTP traffic, no syntactic and protocol-specific strings have been extracted, demonstrating the ability of ASAP to focus on semantics of communication.

3.3 Analysis of Malware Communication

A second application domain for the proposed framework is the automatic analysis of malware communication. While there exist several methods for automatic collection and monitoring of malware [18–21], analysis of monitored malware communication still requires significant manual effort. As a remedy, we apply

Table 2: Templates for honeypot dataset. The templates have been constructed using tokens as basic strings and NMF as factorization.

Communication Templates	Description
1) modez \wedge shellz \wedge csp.txt	Semantics of RFI malware
2) modez \wedge psybnc \wedge csp.txt	—
3) modez \wedge botz \wedge bot.txt	—
4) modez \wedge scannerz \wedge bot.txt	—
5) mosConfig.absolute.path \wedge option \wedge http	Vulnerability (VirtueMart)
6) mosConfig.absolute.path \wedge option \wedge Itemid	—
7) com_virtuemart \wedge show_image_in_imgtag.php ...	—
8) com_virtuemart \wedge export.php \wedge php.txt	—
9) shop_this_skin_path \wedge skin_shop \wedge standard ...	Vulnerability (Technote)
10) board_skin_path \wedge Smileys \wedge http	Vulnerability (GNUBoard)
11) board \wedge skin \wedge http	—
12) write.update.php \wedge files \wedge 1	—
13) write.comment.update.php \wedge files \wedge http	—
14) delete.all.php \wedge admin \wedge zefa.txt	—
15) delete.comment.php \wedge http \wedge fx29id1.txt	—
16) appserv \wedge appserv_root \wedge main.php	Vulnerability (Appserv)
17) _SERVER \wedge DOCUMENT_ROOT \wedge media	Vulnerability (PHP)
18) error.php \wedge dir \wedge 1	Misc. RFI vulnerabilities
19) errors.php \wedge error \wedge php.txt \wedge bot.txt	—
20) administrator \wedge index.php \wedge raw.txt	—
21) admin \wedge include \wedge http	—
22) med.buu.ac.th \wedge com.mylink \wedge stealth ...	Sources of attacks
23) http \wedge med.buu.ac.th \wedge com.mylink \wedge components	—
24) http \wedge www.hfsb.org \wedge sites \wedge 10225 \wedge img	—
25) zerozon.co.kr \wedge eeng \wedge zefa.txt	—
26) http \wedge zerozon.co.kr \wedge photos \wedge count	—
27) http \wedge musicadelibreria.net \wedge footer	—
28) qqe.ru \wedge forum \wedge Smileys	—

the ASAP framework for discovery of typical components in malware communication. In particular, we analyze the communication of 20 malware binaries that has been recorded during repetitive executions of each binary in a sandbox environment [see 22]. Since we do not know, which kind of protocols are contained in the traffic, we apply the ASAP framework using 4-grams as basic strings and extract base directions via the NMF algorithm. From the 20 binaries, we pick *Vanbot* as an example and present the respective communication templates in Table 3 as they particularly emphasize the capabilities of our framework.

First, we observe that the extracted components clearly separate protocol semantics: three components contain IRC related strings (1, 2, 4), while one component contains HTTP data (3). A closer look reveals, that the first two components contain IRC communication typical for *Vanbot*: the malware joins two IRC channels, namely **#las6** and **#ns**, and signifies the start of a TFTP service. The HTTP component in turn comprises update requests, where the malware tries to download updates of itself from different hosts. Interestingly,

Table 3: Templates for communication of a malware binary. The templates have been constructed using 4-grams as basic strings and NMF as factorization.

Communication Templates
1) MODE #las6↔USER b ^ JOIN #las6 ^ 041- Running TFTP wormride...
2) c↔MODE #ns↔USER ^ c +xi↔JOIN #ns ^ ub.28465.com↔PONG :hub.2
3) GET /la1222.exe HTTP/1.0↔Host: zonetech.info ^ /lb3.ex ^ /las1.ex...
4) x↔USER e020501 . . . ^ JOIN &virtu3 ^ NICK bb ^ CK gv ^ R h020

our analysis also extracts an IRC component (4), which contains typical communication of the *Virut* malware, for example, as indicated by the channel name #virtu3. We credit this finding to a co-infection: the malware binary labeled *Vanbot* has been additionally infected by the malware *Virut*, a file infector.

3.4 Anomaly Detection

As final experiment, we evaluate the capabilities of ASAP in the fields of network intrusion detection. Anomaly detection is frequently applied as extension to signature-based intrusion detection systems, such as the Snort [1] and Bro [2] system, as it enables identification of unknown and novel network threats.

For evaluation of intrusion detection performance, we consider two datasets of network payloads: The first dataset (FIRST08) contains HTTP requests monitored at the web server of a research institute during a period of 60 days. The second dataset (FTP03) comprises FTP sessions recorded over 10 days at Lawrence Berkeley National Laboratory [23]. Additionally to this benign data, we inject network attacks into the traffic. The attacks are executed in a virtual environment using common tools for penetration testing, such as Metasploit, and are carefully adapted to match characteristics of the datasets [see 24].

For the experiment, we apply a detection method similar to the work of Rieck and Laskov [25]. A centroid model of normal network payloads is constructed using n-grams, $\mu_{\text{full}} = \frac{1}{N} \sum_{i=1}^N \phi(p_i)$, and used for identifying unusual network content. Additionally, we consider a second model in which the n-grams are refined using communication templates. Formally, after calculating the matrix factorization $X = BC$, we construct this model as follows: $\mu_{\text{reduced}} = B(\frac{1}{N} \sum_{i=1}^N c_i)$, where we calculate the centroid in the lower-dimensional space obtained by the first 20 base directions of NMF. The two models are trained on 1,000 randomly drawn payloads for each data set and anomaly detection is performed on 200,000 randomly chosen HTTP requests and 20,000 FTP sessions respectively.

Results are shown as Receiver Operating Characteristics (ROC) curves in Figure 4 (left). The performance of the full and reduced centroid model is identical on all three datasets. This demonstrates that the base directions identified by NMF capture semantic information of the underlying protocols accurately for detection of anomalies and attacks. Figure 4 (right) details the run-time performance attained by the different models. Reducing the analysis using communication templates provides a significant performance gain over regular anomaly detection. Speed-up factors of up to 15 can be observed and clearly indicate the utility of ASAP as a preprocessing step for anomaly detection.

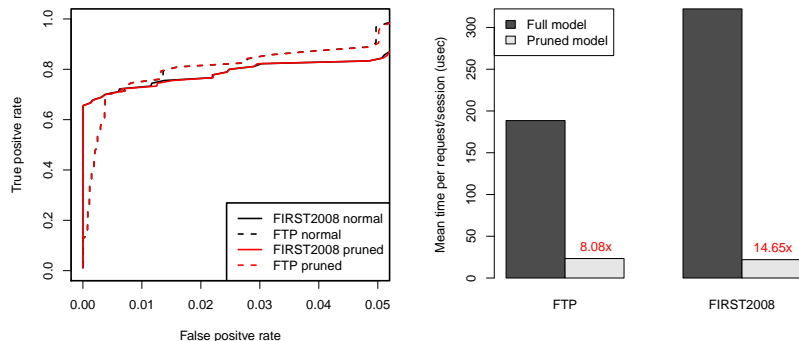


Fig. 4: ROC curves (left) and run-time (right) for network anomaly detection. Models are constructed using 4-grams and NMF for matrix factorization.

4 Related Work

The problem of re-creating a certain structure based on data like network or execution traces has been extensively studied in the domain of protocol reverse engineering. The ultimate goal here is to reconstruct the grammar and also underlying state machines employed during communication. *Discoverer* [10] by Cui et al. works directly on recorded network traces to extract message formats by a combination of clustering of tokens and subsequent merging by sequence alignment. Wondracek et al. [11] are able to create grammar-like structures with the use of dynamic data tainting: a protocol is automatically constructed by monitoring data flow of protocol messages during request serving. This concept is further refined in *Prospex* [12] by Comparetti et al. which also incorporates the inference of the underlying state machine of the communication. All these approaches are orthogonal to the ASAP framework, since they try to extract the underlying syntax of the communication. With the ASAP method we address the extraction of semantics-aware components for a monitored application, which leads to different constraints and methods employed during the analysis.

A natural extension of protocol reverse engineering is to use the newly extracted protocol and communication state automaton and build an automated honeypot service, which is capable of mimicking the monitored application behavior. The work of Leita et al. [19, 20] target the honeyd platform: by examination of a tcpdump, which contains a sample interaction, they are able to generate a honeyd script, which can emulate to a certain extent the monitored application. In line with this work Cui et al. present *RolePlayer* [26], which is able to replay both the client and server side of a given communication. The *Replayer* [27] by Newsome et al. uses methods from program verification to build up a logical sound solution to the replaying problem. The ASAP framework in its current state does not incorporate any communication semantics apart from single message semantics. However, it would be a valuable extension to include communication behavior analysis by exploiting methods of time series analysis inside the ASAP framework.

Dimension reduction methods have been used before to visualize network traffic data [28] or track down traffic anomalies [29, 30]. While Principal Component Analysis and Non-negative Matrix Factorization (NMF) have been studied before, the ASAP framework analyzes the traffic both on the basis of tokens and n-grams in a unified way, which leads to a semantic-driven and easily comprehensible result. Furthermore NMF has also been used for intrusion detection by Wang et al. [31, 32] on system call traces and even for the task of document summarization [33] on a word level. This shows that the basis of the ASAP framework is well established even beyond analysis of structured network data.

5 Conclusions

We have introduced the ASAP framework, a new technique for automatic extraction of communication templates from recorded network traffic. The framework identifies components of network payloads by mapping them to a vector space and determining informative base directions using techniques of matrix factorization. This approach is orthogonal to existing techniques for network analysis based on protocol specifications, as it can be applied for analysis of unknown network protocols as well as network traces containing mixtures of protocols.

Empirically, we have demonstrated the utility of our framework in different security applications. For example, we have been able to automatically analyze data from a network honeypot and identify exploited vulnerabilities and attack sources—a task not attainable by sole means of specification-based analysis. Moreover, we have exemplarily dissected the communication of the malware *Vanbot*, demonstrating the ability of the ASAP framework to discover semantics in network traces of mixed protocols. Finally, we have applied our framework for preprocessing the input of a network anomaly detection method, realizing a speed-up factor of up to 15 while preserving an accurate detection of attacks.

Our analysis shows that base directions generated by our framework can be used to easily extract valuable details from a dataset, where nearly no information is available in advance. The ASAP framework is capable of analyzing network data even without knowledge about the underlying protocols. The flexibility of n-gram analysis paired with the structuring and summarization power of matrix factorization generates semantics-aware components, which enable fast and efficient insights for network traces at hand.

Besides applications presented in this work, the extracted base directions provide good candidates as input for other security methods: signature generation methods may use the concise representation of data as direct basis for constructing precise signatures, automatic construction of honeypot services could greatly profit from semantics-aware representations of communication and the process of forensic network analysis may be accelerated by taking these communication templates as starting points for further investigations. Finally, the incorporation of other matrix factorization methods like sparse NMF [34] or sparse PCA [35] and the evaluation of their respective performance in terms of data explanation and description would be a further extension of the ASAP framework.

Acknowledgments: The authors were supported by the BMBF grant FKZ 01-IS07007A (REMIND) and the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886.

Bibliography

- [1] Roesch, M.: Snort: Lightweight intrusion detection for networks. In: Proc. of USENIX Large Installation System Administration Conference LISA. (1999) 229–238
- [2] Paxson, V.: Bro: A system for detecting network intruders in real-time. *Computer Networks* **31**(23–24) (December 1999) 2435–2466
- [3] Vigna, G., Kemmerer, R.A.: NetSTAT: a network-based intrusion detection system. *Journal of Computer Security* **7**(1) (1999) 37–71
- [4] Offutt, J., Liu, S., Abdurazik, A., Ammann, P.: Generating test data from state-based specifications. *The Journal of Software Testing, Verification and Reliability* **13** (2003) 25–53
- [5] McAllister, S., Kirda, E., Kruegel, C.: Leveraging user interactions for in-depth testing of web applications. In: *Recent Advances in Intrusion Detection (RAID)*. (2008) 191–210
- [6] Abdelnur, H.J., State, R., Festor, O.: Advanced fuzzing in the voip space. *Journal in Computer Virology* **6**(1) (2010) 57–64
- [7] Garfinkel, S.: *Network Forensics: Tapping the Internet*. O’Reilly (2002)
- [8] Moore, D., Shannon, C., Brown, J.: Code-Red: a case study on the spread and victims of an internet worm. In: *Proc. of Internet Measurement Workshop (IMW)*. (2002) 273–284
- [9] Gates, C., McHugh, J.: The contact surface: A technique for exploring internet scale emergent behaviors. In: *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*. (2008) 228–246
- [10] Cui, W., Kannan, J., Wang, H.J.: Discoverer: automatic protocol reverse engineering from network traces. In: *Proc. of 16th USENIX Security Symposium*. (2007) 1–14
- [11] Wondracek, G., Comparetti, P.M., Krügel, C., Kirda, E.: Automatic network protocol analysis. In: *Proc. of Network and Distributed System Security Symposium (NDSS)*. (2008)
- [12] Comparetti, P.M., Wondracek, G., Kruegel, C., Kirda, E.: Prospex: Protocol specification extraction. In: *Proc. of the 30th IEEE Symposium on Security and Privacy*. (2009) 110–125
- [13] Holm, S.: A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* **6** (1979) 65–70
- [14] Rieck, K., Laskov, P.: Linear-time computation of similarity measures for sequential data. *Journal of Machine Learning Research* **9** (2008) 23–48
- [15] Jolliffe, I.: *Principal Component Analysis*. Springer (2002)
- [16] Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10** (1998) 1299–1319
- [17] Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: *Advances in Neural Information Processing Systems 13*. (2000) 556–562
- [18] Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.C.: The nepthes platform: An efficient approach to collect malware. In: *Recent Advances in Intrusion Detection (RAID)*. (2006) 165–184

- [19] Leita, C., Mermoud, K., Dacier, M.: Scriptgen: an automated script generation tool for honeyd. In: Proc. of Annual Computer Security Applications Conference (ACSAC). (2005) 203–214
- [20] Leita, C., Dacier, M., Massicotte, F.: Automatic handling of protocol dependencies and reaction to 0-day attacks with scriptgen based honeypots. In: Recent Advances in Intrusion Detection (RAID). (2006) 185–205
- [21] Bayer, U., Moser, A., Kruegel, C., Kirda, E.: Dynamic analysis of malicious code. *Journal in Computer Virology* **2**(1) (2006) 67–77
- [22] Rieck, K., Schwenk, G., Limmer, T., Holz, T., Laskov, P.: Botzilla: Detecting the "phoning home" of malicious software. In: Proc. of 25th ACM Symposium on Applied Computing (SAC). (2010)
- [23] Paxson, V., Pang, R.: A high-level programming environment for packet trace anonymization and transformation. In: Proc. of ACM SIGCOMM. (2003) 339 – 351
- [24] Krueger, T., Gehl, C., Rieck, K., Laskov, P.: TokDoc: A self-healing web application firewall. In: Proc. of 25th ACM Symposium on Applied Computing (SAC). (2010)
- [25] Rieck, K., Laskov, P.: Detecting unknown network attacks using language models. In: Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA). (2006) 74–90
- [26] Cui, W., Paxson, V., Weaver, N., Katz, R.H.: Protocol-independent adaptive replay of application dialog. In: Proc. of Network and Distributed System Security Symposium (NDSS). (2006)
- [27] Newsome, J., Brumley, D., Franklin, J., Song, D.: Replayer: automatic protocol replay by binary analysis. In: Conference on Computer and Communications Security (CCS). (2006) 311–321
- [28] Patwari, N., Hero, III, A.O., Pacholski, A.: Manifold learning visualization of network traffic data. In: Proc. of the ACM SIGCOMM workshop on Mining network data. (2005) 191–196
- [29] Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. In: Proc. of ACM SIGCOMM. (2004) 219–230
- [30] Ringberg, H., Soule, A., Rexford, J., Diot, C.: Sensitivity of PCA for traffic anomaly detection. In: Proc. of the ACM SIGMETRICS. (2007) 109–120
- [31] Wang, W., Zhang, X., Gombault, S.: Constructing attribute weights from computer audit data for effective intrusion detection. *J. Syst. Softw.* **82**(12) (2009) 1974–1981
- [32] Guan, X., Wang, W., Zhang, X.: Fast intrusion detection based on a non-negative matrix factorization model. *J. Netw. Comput. Appl.* **32**(1) (2009)
- [33] Wang, D., Li, T., Zhu, S., Ding, C.: Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In: Proc. of the 31st ACM SIGIR. (2008) 307–314
- [34] Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.* **5** (2004) 1457–1469
- [35] Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. *Journal of Computational and Graphical Statistics* **15** (2004) 2006–2035