Weierstraß-Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Preprint ISSN 0946 – 8633

ASAP: Automatic semantics-aware analysis of network payloads

Tammo Krueger[1] Nicole Krämer[2] Konrad Rieck[3]

submitted: April 13, 2010

[1] Fraunhofer FIRST
Kekulestr. 7
12489 Berlin
Germany
E-Mail: tammo.krueger@first.fraunhofer.de

[2] Weierstrass Institute
for Applied Analysis and Stochastics
Mohrenstr. 39
10117 Berlin
Germany
E-Mail: nicole.kraemer@wias-berlin.de

[3] Berlin Institute of Technology
Franklinstr. 28/29
10587 Berlin
Germany
E-Mail:konrad.rieck@tu-berlin.de

No. 1502
Berlin 2010

2000 *Mathematics Subject Classification.* 62H25,68M12.

Key words and phrases. dimensionality reduction, computer security, intrusion detection.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
10117 Berlin
Germany

Fax: + 49 30 2044975
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

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 automaticylly extracts semantic components from recorded network traffic. Our method proceeds by mapping network payloads to a vector space and identifying semantic 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. The specification of a protocol defines the basic building blocks of its communication as well as the syntax of its network messages. For the majority of protocols, specifications are publically available and hence can be directly applied for dissecting network data into syntactic components. Moreover, to support analysis of proprietary and unknown protocols, techniques for automatic construction of protocol specifications and state machines have been developed [10–12].

While analysis techniques based on protocol specifications 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 to automatically identify and extract semantic 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 semantic 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 semantic templates of communication. The main contributions of this framework are:

1. *Alphabet extraction for network payloads.* For automatic analysis, we devise a technique for

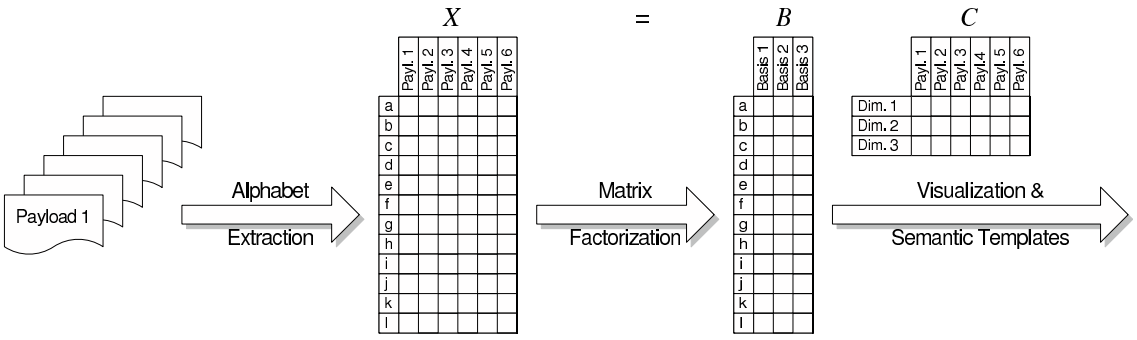


Figure 1: Overview of the ASAP framework.

extracting an alphabet of strings from network payloads. The alphabet concisely characterizes the network traffic and enables a mapping of payloads to a high-dimensional vector space.

2. *Semantic analysis using matrix factorization.* We propose a method for inferring common patterns of usage from network data. The method proceeds by identifying base directions in the vector space of network payloads using concepts of matrix factorization.

3. *Construction of semantic templates.* Discovered base directions are transformed to semantic 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.

The remainder of this paper is structured as follows: The ASAP framework for extraction of semantic templates from network communication is introduced in Section 2. Experiments and empirical results with different security applications are presented in Section 3. Related work to our framework is discussed in Section 4 and the paper concludes with Section 5.

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 high-dimensional 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 semantic template of typical communication content.

2.1 Alphabet Extraction for Network Payloads

A payload p is a string of bytes contained in network communication. A payload may refer to the assembled data of a TCP connection as well as the data of a UDP packet. 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 semantic 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

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

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

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

Alphabet filtering and correlation. Strings within network payloads naturally appear with different frequency, ranging from volatile to constant occurences. 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 with a correlation coefficient of approximately one. Hence, we combine elements of S which co-occur in the analyzed data and thereby further refine our alphabet.

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

$$\phi : P \to \mathbb{R}^{|S|}, \quad \phi : p \mapsto (I(s,p))_{s \in S} ,$$

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. This mapping is illustrated in the following example:

$$\phi(\texttt{aabbab}) \longmapsto \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \begin{matrix} \texttt{aba} \\ \texttt{abba} \\ \texttt{ba} \\ \texttt{bbb}. \end{matrix}$$

The artificial network payload $p =$ `aabbab` is mapped to a 4-dimensional vector space using the alphabet $S = \{\texttt{aba}, \texttt{abba}, \texttt{ba}, \texttt{bbb}\}$. Dimensions associated with strings occurring in the payload yield non-zero assignments, while the remaining dimensions are set to zero.

Depending on the alphabet S the resulting vector space may be spanned by thousands of different dimensions. Efficient computation in such high-dimensional spaces seems intractable at a first glance. However, a payload of length m comprises at most $O(m)$ different tokens and n-grams. As a consequence, the mapping ϕ is *sparse*, that is, the vast majority of dimensions is zero. This sparsity can be exploited to derive linear-time algorithms for extraction and comparison of

vectors. Instead of operating with full vectors, only the non-zero dimensions are considered, where the strings associated with each dimension are maintained in efficient data structures, such as hash tables [14] and Bloom filters [15].

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. This vectorial representation of network data enables us to identify semantic 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 semantic 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

$$X \approx BC = \overbrace{\begin{bmatrix} b_1 & \dots & b_L \end{bmatrix}}^{\text{basis}} \underbrace{\begin{bmatrix} c_1 & \dots & c_N \end{bmatrix}}_{\text{coordinates}}. \tag{1}$$

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 semantic 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. These coordinates can be used for visualization of the data in a low-dimensional space.

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) [16] and Non-negative Matrix Factorization (NMF) [17].

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

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

In this setting the matrix factorization (1) corresponds to a singular value decomposition of X: The L orthonormal basis vectors in B equal the first L left-singular vectors of X, and the coordinates C correspond to the first right-singular vectors of X, multiplied by their singular values.

Non-negative Matrix Factorization. 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 considered to be 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

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

The solution is computed via standard gradient descent techniques. As the basis vectors are not orthogonal, they can share semantic information and induce a hierarchical representation of the matrix X.

2.3 Construction of Semantic 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. As an example we look at the following base vector and the resulting semantic template:

$$\begin{pmatrix} 0.92 \\ 0.82 \\ 0.03 \\ 0.04 \end{pmatrix} \begin{matrix} \texttt{aba} \\ \texttt{abba} \\ \texttt{ba} \\ \texttt{bbb} \end{matrix} \xrightarrow{\geq 0.8} (\texttt{aba} \wedge \texttt{abba}) \tag{2}$$

The re-mapping of the vector results in a conjunction of strings (`aba` $\wedge$ `abba`) similar to a regular expression, where $\wedge$ corresponds to a conjunction operator and the threshold is given by 0.8. Note that we lose the ordering of tokens due to the vectorial representation. 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 `Hos` $\wedge$ `ost` $\wedge$ `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.

This procedure is formalized in pseudo-code in Algorithm 1. The parameter `maxGap` controls the maximal gap allowed for an overlap to be considered for merging. If we consider 4-grams as an example, the gap loop starting in line 6 will first look for an overlap of three (`gap` $= 0$) at the start or end of the tokens, then for an overlap of two (`gap` $= 1$) and so on. It is obvious that high values of `maxGap` can lead to accidental concatenation of n-grams but on the other hand reduce the size of the alphabet.

Algorithm 1 Vector-to-Template Conversion

Require: length(alphabet) = length(weights); maxGap $\in \mathbb{N}^+$

```
1:  function TEXTUALREPRESENTATION(alphabet, weights, maxGap)
2:      representation := []                                          ▷ empty list
3:      L := tokens ordered by weight
4:      while length(L) ≥ 2 do
5:          curToken := L.pop()                  ▷ extract token with highest weight
6:          for (gap := 0; gap ≤ maxGap; maxGap++) do
7:              for (index := 1; index ≤ length(L); index++) do
8:                  token := L[index]
9:                  if overlap(curToken, token, gap) then
10:                     curToken := merge(curToken, token)
11:                     L.remove(token)                  ▷ remove the merged token
12:                     index := 0                       ▷ restart matching process
13:         representation.push(curToken)                ▷ no more overlaps found
14:     representation.extend(L)                         ▷ add any remaining token
15:     return representation
```

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 a toy dataset, which allows us to establish an understanding of how semantic templates are inferred from communication (Section 3.1). We then proceed to real-world applications, where network traces containing malicious communication are analyzed for semantic components, such as exploited vulnerabilities and attack sources (Section 3.2 & 3.3). Finally, we apply our framework for improving the performance of network anomaly detection by limiting the processing of network data using semantic 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 [18]. 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 semantic templates for these base directions.

We present the extracted alphabet in Table 1. The alphabet consists of 8 symbols and corresponds to relevant strings of the underlying web application. Constant and volatile tokens have been filtered, while tokens co-occurring in the communication have been coupled as indicated by the $\wedge$ operator. Note that the alphabet does not contain tokens related to HTTP syntax and thereby

```
GET static/3lpAN6C2.html HTTP/1.1
Host: www.foobar.com
Accept: */*
```
Request for static content

```
GET cgi/search.php?s=Eh0YKj3r3wD2I HTTP/1.1
Host: www.foobar.com
Accept: */*
```
Search query

```
GET cgi/admin.php?action=rename&par=dBJh7hS0r5 HTTP/1.1
Host: www.foobar.com
Accept: */*
```
Administrative request

Figure 2: Example payloads of the artificial dataset.

differs from previous approaches reconstructing protocol grammars and specifications.

Table 1: Extracted alphabet for the artificial dataset. Strings co-occuring in network payloads are coupled using the $\wedge$ operator.

	Symbols		
1)	`static`	5)	`rename`
2)	`cgi`	6)	`move`
3)	`(search.php` $\wedge$ `s)`	7)	`delete`
4)	`(action` $\wedge$ `admin.php` $\wedge$ `par)`	8)	`show`

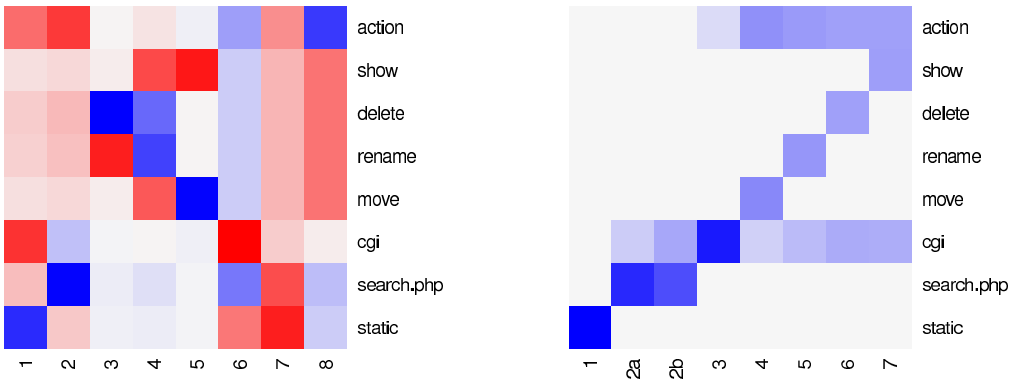
Results for the application of matrix factorization algorithms to the artificial dataset are visualized in Figure 3. For the algorithms PCA and NMF, base directions (matrix B) are shown, where the x-axis details the different directions and the y-axis the contribution of individual alphabet symbols.

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. Semantic templates resulting from the NMF matrix in Figure 3 are presented in Table 2. The templates accurately capture the semantics implemented in the simple web application. A set of 7 templates is constructed which covers static access of web content, search queries and different administrative tasks. Note that two base directions in Figure 3 are identical, resulting in a total of 7 templates. The semantic templates even exhibit hierarchical structure: template 3 resembles a basic administrative request with all following templates being special cases for particular administrative actions.

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



(a) PCA (b) NMF

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

Table 2: Semantic templates extracted for the artificial dataset. The templates have been constructed using tokens as basic strings and NMF for matrix factorization.

	Semantic templates
1)	`static`
2)	`cgi` ∧ `(search.php` ∧ `s)`
3)	`cgi` ∧ `(action` ∧ `admin.php` ∧ `par)`
4)	`cgi` ∧ `(action` ∧ `admin.php` ∧ `par)` ∧ `move`
5)	`cgi` ∧ `(action` ∧ `admin.php` ∧ `par)` ∧ `rename`
6)	`cgi` ∧ `(action` ∧ `admin.php` ∧ `par)` ∧ `delete`
7)	`cgi` ∧ `(action` ∧ `admin.php` ∧ `par)` ∧ `show`

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*[1]. 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 semantic templates using the base direction identified by NMF from the embedded HTTP payloads. The templates are shown in Table 3. Note that 12 templates have been omitted as they contain redundant and unspecific information.

The extracted 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

[1]Glastopf Web Application Honeypot – http://glastopf.org

Table 3: Semantic templates for honeypot dataset. The templates have been constructed using tokens as basic strings and NMF for matrix factorization.

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

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

	Semantic 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 /lal222.exe HTTP/1.0↔Host: zonetech.info ∧ /lb3.ex ∧ /las1.ex...`
4)	`x↔USER e020501 . . . ∧ JOIN &virtu3 ∧ NICK bb ∧ CK gv ∧ R h020`

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 semantic 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

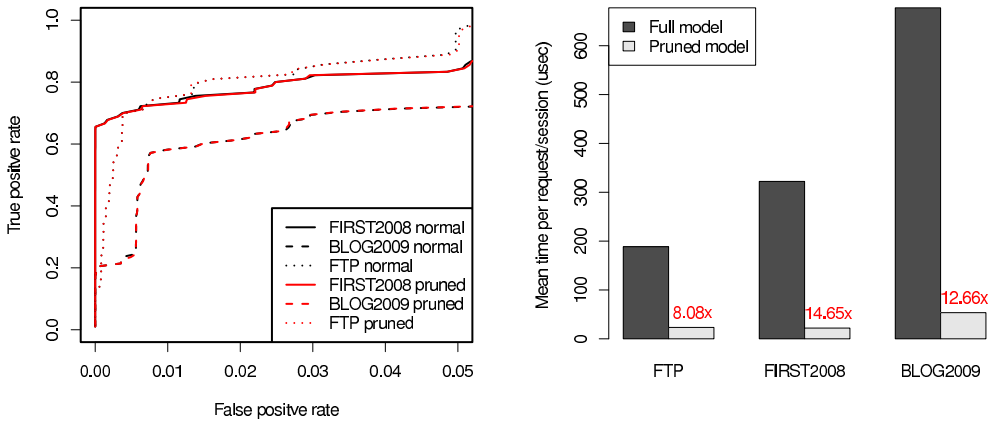
A second application domain for the proposed framework is the automatic analysis of malware communication. While there exists several methods for automatic collection and monitoring of malware [19–23], analysis of monitored malware communication still requires significant manual effort. As a remedy, we apply the ASAP framework for discovery of typical semantic 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 24]. 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 semantic templates in Table 4 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, 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.

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 components, which enable fast and efficient insights for network traces at hand. Note that the general methodology of the ASAP framework itself is not limited to text-based protocols. Although results from the experiments with *Vanbot* yield mainly textual patterns, the underlying extraction of n-grams is capable to operate on binary data as well.

3.4 Anomaly Detection

As final experiment, we evaluate the capabilities of our framework for the application of network intrusion detection. Techniques of anomaly detection are frequently applied as extension to signature-based intrusion detection systems, such as the popular Snort [1] and Bro [2] system, as



(a) ROC curves

(b) Run-time performance

Figure 4: ROC curves and run-time for network anomaly detection. The templates have been constructed using 4-grams as basic strings and NMF for matrix factorization.

they enable identification of unknown and novel network threats.

For evaluation of intrusion detection performance, we consider three larger datasets of network payloads: The first dataset (FIRST08) contains HTTP requests monitored at the web server of our research institute during a period of 60 days. The second dataset (BLOG09) contains requests of several web blogs running the popular platform WordPress and spans 33 days of traffic. The third dataset (FTP03) comprises FTP sessions recorded over 10 days at Lawrence Berkeley National Laboratory [25]. 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 26].

For the experiment, we apply a detection method similar to the work of Wang et al. [27; 28]. 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 semantic templates. Formally, after calculating the matrix factorization $A = BC$, 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 4a. 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 sufficient for detection of anomalies and attacks. Figure 4b details the run-time performance attained by the different models. Reducing the analysis using semantic templates provides a significant performance gain over regular anomaly detection. Speed-up factors between 8–15 can be observed and clearly indicate the utility of semantics-aware analysis as a preprocessing step for anomaly detection.

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 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. [21; 22] 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* [29], which is able to replay both the client and server side of a given communication. The *Replayer* [30] 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 [31] or track down traffic anomalies [32; 33]. 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. [34; 35] on system call traces and even for the task of document summarization [36] 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 semantic templates from recorded network traffic. The framework identifies semantic components of network payloads by mapping payloads 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. Moreover, the ASAP framework can be coupled with specification-based techniques to allow for joint analysis of syntactic and semantic components in network communications.

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 between 8–15 while preserving an accurate detection of attacks and anomalies.

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 semantic templates as starting points for further investigations. Finally, the incorporation of other matrix factorization methods like sparse NMF [37] or sparse PCA [38] and the evaluation of their respective performance in terms of data explanation and description would be a further extension of the ASAP framework.

References

[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 interactioncs for in-depth testing of web applications. In: Recent Advances in Intrusion Detection, 11th International Symposium, RAID 2008, Boston, USA, Proceedings. (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: DIMVA. (2008) 228–246

[10] Cui, W., Kannan, J., Wang, H.J.: Discoverer: automatic protocol reverse engineering from network traces. In: SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, Berkeley, CA, USA, USENIX Association (2007) 1–14

[11] Wondracek, G., Comparetti, P.M., Krügel, C., Kirda, E.: Automatic network protocol analysis. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008. (2008)

[12] Comparetti, P.M., Wondracek, G., Kruegel, C., Kirda, E.: Prospex: Protocol specification extraction. In: SP '09: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, Washington, DC, USA, IEEE Computer Society (2009) 110–125

[13] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6** (1979) 65–70

[14] Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science **267**(5199) (1995) 843–848

[15] Wang, K., Parekh, J., Stolfo, S.: Anagram: A content anomaly detector resistant to mimicry attack. In: Recent Adances in Intrusion Detection (RAID). (2006) 226–248

[16] Jolliffe, I.: Principal Component Analysis. Springer (2002)

[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] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard) (June 1999) Updated by RFC 2817.

[19] Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.C.: The nepenthes platform: An efficient approach to collect malware. In: Recent Adances in Intrusion Detection (RAID). (2006) 165–184

[20] Willems, C., Holz, T., Freiling, F.: CWSandbox: Towards automated dynamic binary analysis. IEEE Security and Privacy **5**(2) (2007) 32–39

[21] Leita, C., Mermoud, K., Dacier, M.: Scriptgen: an automated script generation tool for honeyd. In: ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference, Washington, DC, USA, IEEE Computer Society (2005) 203–214

[22] 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, 9th International Symposium, RAID 2006, Hamburg, Germany, September 20-22, 2006, Proceedings. (2006) 185–205

[23] Bayer, U., Moser, A., Kruegel, C., Kirda, E.: Dynamic analysis of malicious code. **2**(1) (2006) 67–77

[24] Rieck, K., Schwenk, G., Limmer, T., Holz, T., Laskov, P.: Botzilla: Detecting the phoning homeöf malicious software. In: Proc. of 25th ACM Symposium on Applied Computing (SAC). (March 2010) (to appear).

[25] Paxson, V., Pang, R.: A high-level programming environment for packet trace anonymization and transformation. In: Proc. of Applications, technologies, architectures, and protocols for computer communications SIGCOMM. (2003) 339 – 351

[26] 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). (March 2010) (to appear).

[27] Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In: Recent Adances in Intrusion Detection (RAID). (2004) 203–222

[28] Wang, K., Cretu, G., Stolfo, S.: Anomalous payload-based worm detection and signature generation. In: Recent Adances in Intrusion Detection (RAID). (2005)

[29] Cui, W., Paxson, V., Weaver, N., Katz, R.H.: Protocol-independent adaptive replay of application dialog. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, USA. (2006)

[30] Newsome, J., Brumley, D., Franklin, J., Song, D.: Replayer: automatic protocol replay by binary analysis. In: CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, New York, NY, USA, ACM (2006) 311–321

[31] Patwari, N., Hero, III, A.O., Pacholski, A.: Manifold learning visualization of network traffic data. In: MineNet '05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data, New York, NY, USA, ACM (2005) 191–196

[32] Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. In: SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA, ACM (2004) 219–230

[33] Ringberg, H., Soule, A., Rexford, J., Diot, C.: Sensitivity of pca for traffic anomaly detection. In: SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, New York, NY, USA, ACM (2007) 109–120

[34] 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

[35] Guan, X., Wang, W., Zhang, X.: Fast intrusion detection based on a non-negative matrix factorization model. J. Netw. Comput. Appl. **32**(1) (2009) 31–44

[36] Wang, D., Li, T., Zhu, S., Ding, C.: Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2008) 307–314

[37] Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. J. Mach. Learn. Res. **5** (2004) 1457–1469

[38] Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. Journal of Computational and Graphical Statistics **15** (2004) 2006–2035