Adaptive Detection of Covert Communication in HTTP Requests

Guido Schwenk
Machine Learning Group
Technische Universität Berlin
Berlin, Germany

Konrad Rieck
Machine Learning Group
Technische Universität Berlin
Berlin, Germany

***Abstract*—The infection of computer systems with malicious software is an enduring problem of computer security. Avoiding an infection in the first place is a hard task, as computer systems are often vulnerable to a multitude of attacks. However, to explore and control an infected system, an attacker needs to establish a communication channel with the victim. While such a channel can be easily established to an unprotected end host in the Internet, infiltrating a closed network usually requires passing an application-level gateway—in most cases a web proxy—which constitutes an ideal spot for detecting and blocking unusual outbound communication.**

This papers introduces DUMONT, a system for detecting covert outbound HTTP communication passing through a web proxy. DUMONT learns profiles of normal HTTP requests for each user of the proxy and adapts to individual web surfing characteristics. The profiles are inferred from a diverse set of features, covering the structure and content of outbound data, and allowing for automatically identifying tunnels and covert channels as deviations from normality. While this approach does not generally rule out sophisticated covert communication, it significantly improves on state-of-the-art methods and hardens networks against malware proliferation. This capability is demonstrated in an evaluation with 90 days of web traffic, where DUMONT uncovers the communication of malware, tunnels and backdoors with few false alarms.

***Keywords*-Network Security, Anomaly Detection, Machine Learning, Covert Channels, Malicious Software**

I. INTRODUCTION

Computer networks face a wide variety of threats from malicious software (*malware*). Just a few years ago, malicious software could be categorized into a few basic classes. Now we are confronted with a plethora of malicious tools developed by an underground economy for monetary gains [e.g., 1–3]. This malware is characterized by versatile functionality and the capability to take numerous routes to a victim, ranging from malicious documents and shortened links to drive-by downloads and targeted attacks. In practice, detecting and eliminating all these infection vectors has proven to be an intractable task and thus millions of hosts in the Internet are plagued by malicious software.

Once compromised, infected machines are regularly misused for illegal activities, such as gathering personal data, distributing spam messages or conducting attacks against other hosts. All these activities inherently require establishing a communication channel that enables the attacker to retrieve data and control the infected system. Such a channel can be trivially established to an unprotected host, for example, by directly sending network packets as performed by the trojans Storm and Nugache [4, 5]. As a result, a large body of research has studied methods for detecting direct communication with infected hosts [e.g. 6–9]. However, enterprise and government networks are often shielded from the Internet by an application-level gateway—typically in form of a packet filter and a web proxy—and thus no direct communication with infected machines can be established. In this setting, the malicious software is required to tunnel its communication through the web proxy and there is a need for methods capable of detecting tunneled and covert communication in HTTP.

In this paper, we introduce DUMONT, an anomaly detection system for identifying tunneled and covert communication passing through a web proxy. DUMONT learns profiles of normal HTTP requests for each user of the proxy and thereby adapts to the individual web surfing characteristics of each user. The individual profiles are inferred from a diverse set of features, covering the structure and content of outbound data. Using these profiles, tunnels and covert communication of malicious software can be identified as deviations from normality, where respective requests can be put on hold and further investigated before leaving the network. Similarly, DUMONT can be applied for analysis of suspicious files in a sandbox, where it can detect unusual web traffic, for instance, when a spyware program transfers gathered data to a remote host.

Detecting covert channels in the general case is a very ambitious task and clearly DUMONT can not spot arbitrarily sophisticated covert communication, for example, using the timing of requests for encoding information. However, the involved implementation and low transmission rates of such advanced channels render them less attractive for adversaries. In practice DUMONT significantly improves on the detection capabilities of related methods such as WEBTAP [10] and raises the bar for malware authors to comprise networks. In an empirical evaluation with 90 days of web traffic from six users, DUMONT allows to identify the majority of malicious software, tunnels and backdoors with a false-positive rate of 0.35%, whereas the rule-based method WEBTAP suffers from over 3% false-positives due to the dynamics of web traffic.

The rest of this paper is structured as follows: In Section II we discuss the dynamics of HTTP. The DUMONT system and underlying learning techniques are presented in Section III and evaluated in Section IV. Section V presents related work and Section VI concludes this paper.

II. DYNAMICS OF HTTP COMMUNICATION

The HTTP protocol features a diversity of properties, exploitable to learn something about a user's communication behavior. While most of them look static at a first glance, they show a rather dynamic behavior in practice.

When analyzing for example, which web sites a set of users visits during a defined time period, one might consider creating a whitelist of benign web sites sufficient for stopping outbound communication to malicious sites. This assumption is unrealistic, as Figure 1(a) illustrates. Formerly unknown (i.e. first-time seen) benign web sites do always occur, be it through the evolution of the Internet or just through the normal web behavior of the user. Furthermore previously benign web sites might have been infected recently, making them no longer viable for a benign whitelist. For those reasons we decide not to learn concrete web site addresses, but to model them indirectly using machine learning.

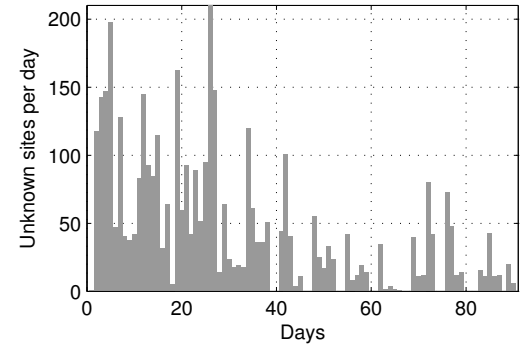
Another dynamic behavior can be observed in the occurrences of HTTP headers. For example, creating a simple whitelist of the appearances of the header *User-Agent* for individual users can provide a means to detect deviations and suspicious combinations (e.g. of the operating system and the web browser). However, this static approach is not sufficient as well, as Figure 1(b) depicts. Previously unseen *User-Agents* occur all the time, be it due to changing tools on the client side or simple version changes in the different web clients. A more indirect method of modeling HTTP requests is necessary here as well.

Another question is, whether adaptive learning on the data of individual users provides advantages over learning on an agglomerated dataset of several users together. As Figure 1(c) illustrates for the distribution of a single feature of HTTP traffic, namely the lengths of the requests, the same feature may show a different statistical behavior for each user. As learning a representative model of normality requires those features to have consistent statistics, learning on data of individual users is preferable to learning on data of all users combined.

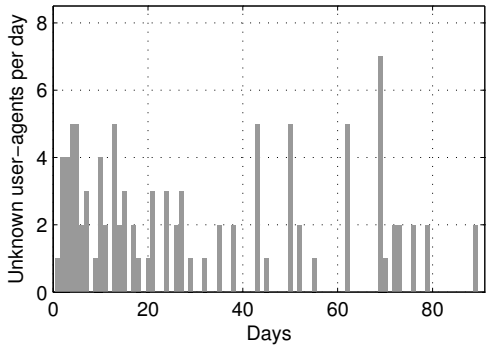
These three examples demonstrate that the dynamics of HTTP communication can hardly be tackled by rule-based methods, such as WEBTAP [10]. Hence, we employ a learning-based approach to detection of covert communication in HTTP, capable of adapting to the individual characteristics of each user.

III. THE DUMONT SYSTEM

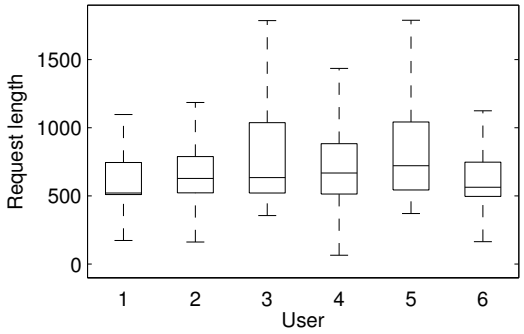
In the following, we present the design of our system DUMONT and its inner workings. The selection of features



(a) Unknown sites per day



(b) Unknown user-agents per day



(c) Distribution of request lengths

Figure 1. Examples of the dynamics in HTTP communication. The communication has been recorded from six users over a period of 90 days. Consult Section IV-A for further details on the data.

of outbound HTTP traffic is discussed in Section III-A, while the necessary learning method as well as the design of the detector are introduced in Section III-B. The concrete necessities and technical details of the operation are then laid out in Section III-C.

A. Features of HTTP Requests

According to RFC2616 [11] an HTTP request starts with a method, e.g. `GET` or `POST`. A method requires an URI, which may include pairs of parameters and values. After the URI, HTTP headers are defined, again consisting of pairs of parameters and values. If a `POST` request is triggered, typically a body of data concludes the request. Additional

to features of this data, each request is triggered at a certain point in time, whose features can be stored as well. Based on those elements of HTTP requests we extract 17 descriptive features grouped in 4 semantic sets.

In networks secured with DUMONT, the use of the HTTP method `CONNECT` is to be restricted, as this method implements a standard tunnel protocol. Allowing an unmonitored use of such a communication channel does undermine the objective of DUMONT. Therefore requests using that method are not taken into account.

Length features: The set of length features is depicted in Table I. It describes length values of different parts of the request, such as URI and body, for later detecting deviations from those values.

Table I
LENGTH FEATURES OF HTTP REQUESTS.

Feature	Description
l_1	Length of request
l_2	Length of URI
l_3	Total length of URI parameters
l_4	Total length of headers
l_5	Length of request body

Structural features: The set of structural features is shown in Table II. It contains values describing the structure of an HTTP request by statistical measures, such as the average length of URI parameter names or header values. This perspective allows identifying outbound data otherwise hidden through distribution over different headers or parameters.

Table II
STRUCTURAL FEATURES OF HTTP REQUESTS.

Feature	Description
s_1	Average length of URI parameter names
s_2	Average length of URI parameter values
s_3	Average length of header names
s_4	Average length of header values

Entropy features: The set of entropy features, depicted in Table III, contains entropy values for different bit widths. These values allow an estimation of the information content in the analyzed request, where the different bit widths cover the request content at different granularity.

Table III
ENTROPY FEATURES OF HTTP REQUESTS.

Feature	Description
e_1	8-bit entropy of request
e_2	16-bit entropy of request
e_3	24-bit entropy of request
e_4	32-bit entropy of request

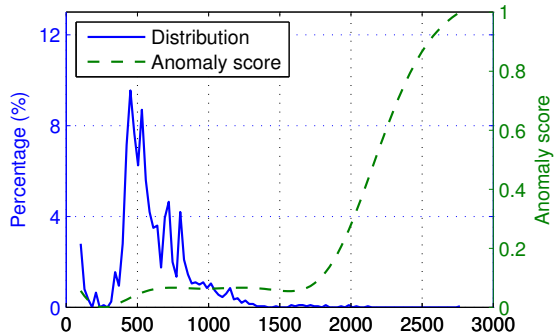
Temporal features: The set of temporal features is illustrated in Table IV. These features enable the analysis of temporal traffic characteristics and help to spot unusual communication activity.

Table IV
TEMPORAL FEATURES OF HTTP REQUESTS.

Feature	Description
t_1	Number of requests in last minute
t_2	Number of outbound bytes in last minute
t_3	Hour of HTTP request
t_4	Week day of HTTP request

B. Anomaly Detection

Our system DUMONT makes use of a standard learning technique—the *One-Class SVM* [12, 13]—for learning a model of normality for the different HTTP features. Formally, a One-Class SVM describes a sphere. This sphere encloses given data with a minimal volume. Anomalies are detected through their distance from the center of the learned sphere, resulting in a high anomaly score. To compensate outliers and noise, as well as to optimize false-positive and detection rates, a soft margin is used. This way not all normal data points are required to reside within the sphere. By using specialized functions, so-called kernels, the sphere can be embedded into a high-dimensional feature space, facilitating the modeling of more complex structures with non-linear representations. In our setup, we use Gaussian kernels [14] to achieve this.



(a) Length of requests

Figure 2. Frequency distribution and anomaly score of request length.

A non-linear model for normality is illustrated in Figure 2, on the example of the length of requests. The left y-axis shows the frequency of different request lengths for one user and the right y-axis shows a function of the anomaly score computed using a One-Class SVM with Gaussian kernels. In principle longer requests cause a higher anomaly score. However, the local minimum at a request length of 1.600 byte demonstrates the advantage of a non-linear representation to model ranges of normality more subtle than simple upper and lower bounds can do.

Hierarchical detection layers: Another important concept realized in DUMONT is the combination of individual detectors in hierarchical layers. The trained models for individual features respectively (*detection layer 0*) provide the capa-

bility to detect covert channels reflected in single features. However, by training models on the combined features of the four feature sets (*detection layer 1*), the detection capability can be further increased to also identify anomalies in the combination of features. This way for example, anomalous requests can be detected, which are normal in both length and entropy, but anomalous in their combination.

The concept is illustrated in Figure 3, depicting *detection layer 0* and *detection layer 1*, as well as *detection layer 2*, which consists of a trained model for all individual features combined, enabling even more synergistic effects.

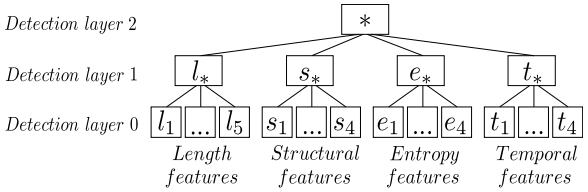


Figure 3. Hierarchical detection layers.

When analyzing an unclassified request, all of those detectors are applied, such that each detector decides, whether the request is normal or anomalous. DUMONT classifies a request as anomalous, if at least one detector triggers an alarm. This approach provides a maximal detection rate, though increasing the false-positive rate as well. The alternative—classifying a request as anomalous when at least n detectors trigger alarms—is no option here, as it allows malware to cover its communication by hiding information within the features of $n-1$ detectors.

C. *Training* DUMONT

For the training of DUMONT several steps are necessary. Initially the normal data is split into training, validation and testing datasets, maintaining the temporal order of the requests. After training and selecting a suitable model for each detector, a sample of malicious communication is used to calibrate the hierarchical detectors of DUMONT.

Model selection: Training a model of normality with a One-Class SVM requires optimizing two parameters, namely the width of the Gaussian kernel and the "softness" of the One-Class SVM. For each combination of these parameters a different SVM model is obtained. From those models a suitable one is picked by the following heuristic:

1) A threshold is set to define an upper bound on the desired false-positive rate on the training dataset. Of the calculated models the one with the highest false-positive rate below this threshold is selected.
2) If models with identical false-positive rates occur, the one with the highest number of support vectors is selected, as this corresponds to the best adaptation to the training data.

Automatic calibration: After selecting a suitable model for each detector on the normal training dataset, the representation quality of the model is further optimized by calibrating the radius of the soft margin of the SVM on the validation dataset using a Receiver Operating Characteristic (ROC) curve. To generate a ROC, we process the validation data and a sample of malicious requests with DUMONT using different thresholds (radius). In principle, a good threshold corresponds to the point closest to $(0.0, 1.0)$ in the ROC. Since we aim primarily for a low false-positive rate however, the use of this point is not recommended, as it corresponds to a one-to-one ratio of false-positive and detection rate. The focus of DUMONT has to be a low false-positive rate, because each of the detectors in Figure 3 is able to trigger an alarm if it detects an anomaly. As a result the individual false alarms of the detectors are accumulated, rendering it most important to keep the false-positive rate of each detector low. Though this results in generally smaller detection rates as well, the negative impact on the overall detection rate is small, since the correctly classified anomalies of the individual detectors are accumulated as well.

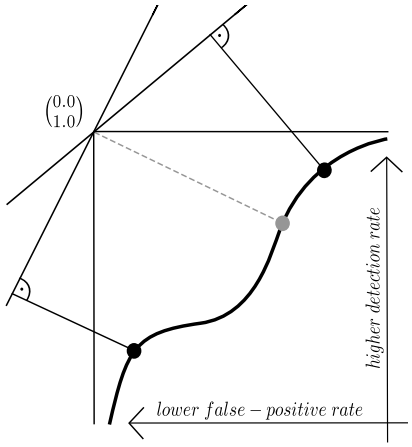


Figure 4. Calibrating a detector using the ROC curve.

To assure a low false-positive rate for each of the individual detectors, the method illustrated in Figure 4 is implemented. Thresholds corresponding to a suitable ratio of false-positive and detection rate are found at the ascending gradients before any local plateau of the ROC curve. They are retrieved by positioning a linear function in $(0.0, 1.0)$ and selecting the point which is closest to that function. Two examples of such points are depicted in Figure 4 as black dots. While both of them have a good ratio of false-positive and detection rate, only the left one, selected by a linear function with a higher gradient, yields a low false-positive rate. As a consequence, we pick the linear function with the steepest gradient to determine the threshold for each detector in our system.

D. Limiting Evasion

One problem of anomaly detection in general are evasion attacks that aim at poisoning the learning data [15, 16]. If an adversary knows the distribution of the features of the normal requests, he can tune his malware to generate seemingly normal traffic with high anomaly scores. Such data points near the margin of the One-class SVM can shift the model towards any direction desired by the attacker, resulting in a setup where once anomalous data is now classified as normal. Fortunately, different methods have been developed to increase the robustness of anomaly detection and to minimize the influence of an adversary. In particular, the techniques of bootstrapping [17] and sanitization [18] can be applied to adjust and filter out anomalous data from the training corpus.

IV. Empirical Evaluation

For the empirical evaluation, datasets of normal and malicious HTTP requests have been collected and used for different experiments. The datasets and the results of those experiments, as well as a comparison with a state-of-the-art approach, are illustrated in this section.

A. Evaluation Data

For collecting normal outbound HTTP requests, a dedicated proxy server has been set up at our institute. After discussing considerations of data privacy, six users were willing to use the proxy server for web access. In the resulting traffic dumps of outbound HTTP traffic, the IP addresses of all users have been pseudonymized. Statistics of the resulting data set of 90 days is presented in Table V. In total the six users generated 143MB of HTTP requests, consisting of 182.996 requests with altogether 173 days of usage and 5.272 requests per day.

For collecting malicious HTTP communication data, different sources have been used. In particular, samples of malicious software have been obtained from the Internet Early Warning system [19] hosted at the University of Mannheim and different honeypots running at our institute. In total a dataset of 2.765 malicious executable files and PDF documents has been collected for our experiments.

To retrieve the kind of communication data relevant for our problem, a small virtual network has been set up, where the binaries and PDF documents are automatically executed in a virtual machine running Windows XP, providing each of them a time frame of 15 minutes to get active. The connections triggered by the malware are redirected to a virtual machine simulating the Internet using TRUMANBOX [20]. To model the desired network layout, a HTTP proxy has been included in our setup. Any HTTP connections from the Windows machine had to find and use that proxy, using information found in the preferences and registry entries prepared on the Windows machine. Of the 2.765 malicious files only 695 have been capable of doing this, whereas several common malware families failed to correctly communicate with the web proxy and would not have been able to establish a communication outside a closed network.

Table VI
STATISTICS OF MALICIOUS WEB TRAFFIC

	# Sessions	# Requests
Malicious software	695	12,899
HTTP tunnels	11	164
Web backdoors	12	345

Besides malicious software, there also exist public tools for establishing outbound communication channels to a system. In particular, we include the web backdoors MATAHARI[1] and RWW-SHELL[2] in our experiments for creating covert communication. Both backdoors have been run with a polling interval of 10 seconds, executing 10–20 shell commands in each session. Moreover, we consider the common tunnel software HTTPTUNNEL[3] for tunneling various traffic through the web proxy.

It is noteworthy that we also experimented with other software for establishing tunneled communication, such as CORKSCREW, SKYPE and TEAMVIEWER. However, these tools make use of the `CONNECT` method for communication and have been excluded from our experiments, as they can be trivially detected and blocked. The statistics of the resulting dataset of malicious HTTP requests are presented in Table VI.

B. Evaluation Setup

To conduct the training of DUMONT in our experiments, we choose the temporally first third of the normal data. The trained models are then validated and calibrated on the temporally second third of the normal data and the validation partition of the malicious data. For realistic experimental results, the validation is repeated ten times, each time with a newly randomized set of validation data of the malicious dataset. For testing, the detectors are applied on the remaining temporally last third of the normal dataset, as well as the malicious test data, remaining in each of the randomizations. Due to that approach the final false-positive and detection rates are presented as the average values of those ten repetitions.

The features t_1–t_4 and t_* are not included in the evaluation. In practice they help detecting malicious outbound traffic at unusual times or with an unusual request frequency. Due to our methods of collecting malicious requests, however, we could not fully test that, because for collecting malicious communication data the binaries and infected documents have been executed over night and weekend as well. Due to the resulting time stamps the corresponding time

[1] A Simple Reverse HTTP Shell, http://matahari.sourceforge.net
[2] Placing Backdoors Through Firewalls, http://www.thc.org/releases.php
[3] GNU HTTP Tunnel, http://www.nocrew.org/software/httptunnel.html

Table V
WEB TRAFFIC OF SIX USERS RECORDED OVER 90 DAYS.

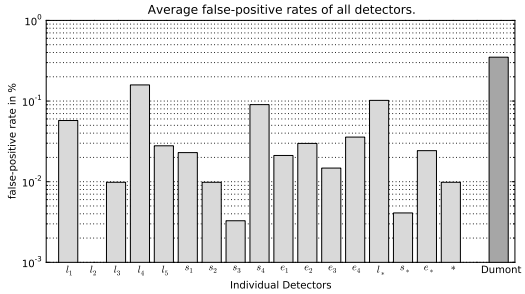
	User 1	User 2	User 3	User 4	User 5	User 6	Total
Number of Requests	116,565	29,723	18,834	9,882	4,001	3,991	182,996
Data Volume	85 Mb	36 MB	11 MB	6 MB	2 MB	3 MB	143 MB
Active Days	50	43	52	10	8	10	173
Requests per Active Day	2,331	691	362	988	500	399	5,272

features contain artifacts and thus are easily distinguishable from benign communication.

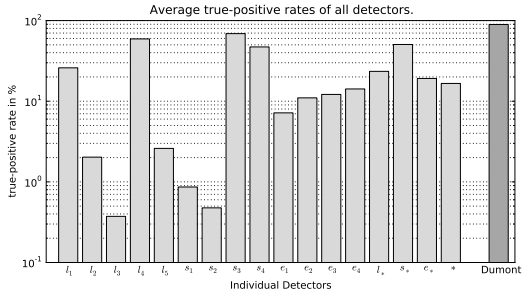
C. System Performance

The detection and false-positive rates of DUMONT for each user are presented in Table VII. Applied on the traffic of tunnels, web backdoors and malicious software, DUMONT performs decently with detection rates of 100.0%, 94.3% and 89.3% respectively. The average false-positive rate reaches a value of 0.35%. While the detection rates of tunnels remain static among all users, the detection of backdoors and malware is strongly user-dependent due the variance of HTTP traffic. This variance is also the reason why rule-based methods are limited in detecting these covert channels, as we will see in Section IV-D.

In principle, each hierarchical detector contributes to the final detection performance of DUMONT. Covert communication can be spotted in all of the features extracted from HTTP requests, as can be seen in Figure 5, which depicts the average contribution of the individual detectors on the false-positive and detection rates.



(a) Contribution to false positives.



(b) Contribution to true positives

Figure 5. Contribution of each detector to the detection performance.

The false positives triggered by DUMONT are mainly caused by large data uploads and most notably cookies. While both types of requests are generally useful for interacting with the Internet, it is obvious that especially in our scenario they represent an inherent risk. Both methods are used to send a bigger and often encrypted amount of data to a server located outside of the protected network, which is what our system is designed to prevent. Such false positives could even be interpreted as true positives when found in a network with stricter security protocols. Since this is not our initial assumption, we keep them as false positives.

DUMONT is implemented in Java, with no special performance optimization. On a single core of an Intel Core2 Duo with 3.00GHz, the whole normal dataset, containing the requests of six users of 90 days, can be processed (i.e. extracting the features and applying the trained detectors) within five minutes. This equates to a run-time performance of approximately 1.300 requests per second.

D. Comparative Evaluation

In our second experiment, we compare the detection performance of DUMONT to the tool WEBTAP [10], that detects covert communication using a mix of filters, trained rules and threshold values. The comparison is conducted on the dataset introduced in Section IV-A.

In terms of the detection rate, WEBTAP performs perfectly and identifies 100% of the traffic of tunnels, web backdoors and malicious software. As discussed in the previous section (Table VII), DUMONT performs slightly worse. However, in terms of false-positive rate, DUMONT significantly outperforms WEBTAP, as WEBTAP flags 3.6% of the requests as covert communication and thus generates more than ten times more false alarms in our experiments. These false alarms are due to the dynamics of HTTP traffic and can be attributed to changing header names and values.

To further illustrate this shortcoming, we investigate the influence of header changes on the overall detection performance. To this end, we change the *User-Agent* of each malicious request to the one most frequently used by the tested user. As a result, all malicious requests contain "faked" user agents. This method of masquerading malicious web traffic can be implemented into malware with only little effort. In this setting, the detection rate of WEBTAP drastically decreases from 100% to 3.7%. The performance of DUMONT, however, as presented in Table VII decreases only slightly from 89% to 82%. Obviously, the detection of

Table VII
DETECTION PERFORMANCE OF DUMONT

	User 1	User 2	User 3	User 4	User 5	User 6	Average
Detection rates							
HTTP tunnels	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
HTTP backdoors	88.0 %	100.0 %	100.0 %	84.0 %	94.0 %	100.0 %	94.3 %
Malware	79.9 %	96.5 %	98.7 %	80.8 %	82.1 %	97.6 %	89.3 %
Malware (faked UA)	67.5 %	94.3 %	89.4 %	73.4 %	69.7 %	97.9 %	82.0 %
False-positive rates							
Benign web traffic	0.14 %	0.39 %	0.23 %	1.18 %	1.12 %	4.05 %	0.35 %

WEBTAP strongly depends on static header fields and thus can be easily thwarted. The comparison shows strikingly that already a little effort on the attackers side results in a security loss if the employed detection method can not be adapted to the individual characteristics and dynamics of HTTP communication.

V. RELATED WORK AND LIMITATIONS

Closest to our system DUMONT is the work of Borders and Prakash [10], which derive rule-based techniques for detecting covert communication in web traffic. While effective in different settings, these approaches assume that HTTP communication remains static over time and that features extracted from requests are stationary. With the adoption of HTTP as a generic communication protocol for many applications, these assumptions fail in practice and a more adaptive approach for modeling normality is needed.

The most common approach for detecting malware is the use of signatures. While in the past mainly focusing onto inbound traffic [e.g. 21–23], recent work has studied detecting outbound malicious HTTP traffic via automatically generated signatures [8, 9, 24]. In the case of a secured network, where malware tries to establish a covert outbound channel to leak specific information, an adversary surely will avoid using known patterns of malware communication and thus signature-based detection is not effective. Finally, various research on detecting the communication of bot networks [e.g. 6, 7] is related to our approach and makes use of similar concepts. Yet this work focuses on identifying direct communication with end hosts and is not suitable for determining anomalous requests in a web proxy.

A different strain of research has studied techniques for circumventing the leakage of confidential data by monitoring sensitive data in host systems [25, 26]. Although effective in practice, these approaches work only on a system where memory access can be monitored. By contrast, DUMONT can be directly deployed in a network without modifying the operating system of connecting hosts.

One of the limitations of detecting covert channels is based on general coding theory [27]. For HTTP, a good example is described by Feamster et.al. [28], where one party monitors accesses to certain benign web sites, while another party accesses those web site in a specifically arranged pattern. The transferred information is hidden within that pattern and therefore completely undetectable. Though fortunately such approaches limit the bandwidth of information to a minimum, they remain problematic, especially considering long term information leakage through insiders.

VI. CONCLUSIONS

In this paper, we have presented a novel approach for detecting covert and tunneled communication passing through a web proxy. Our system DUMONT builds on hierarchical detectors that can identify anomalous communication in various features of HTTP requests. By using machine learning techniques, we are able to apply DUMONT to the individual traffic of each user and thus can adapt to particular web surfing characteristics automatically. Empirically, we can demonstrate that this setting provides a better detection performance than current static approaches, where DUMONT can identify the communication of malicious software, tunnels and backdoors with only few false alarms.

An interesting topic for future work is to further extend the set of features. For example in a hybrid approach, features from keystrokes or mouse movement [29] might be added to our system to achieve an improved detection performance. Similarly, daily bandwidth limitations as used in WEBTAP could easily be implemented to complement our approach. Finally, the integration of DUMONT into different network environments, e.g. for mobile devices or sensors, may provide perspectives for network-based detection of unknown malicious activity.

VII. ACKNOWLEDGEMENTS

The authors gratefully acknowledge funding from the German Federal Office for Information Security (BSI) and the German Federal Ministry of Education and Research (BMBF) under the project PROSEC (FKZ 01BY1145). The authors would like to thank Klaus-Robert Müller for fruitful discussions and support.

REFERENCES

[1] J. Franklin, V. Paxson, A. Perrig, and S. Savage, "An Inquiry Into the Nature and Causes of the Wealth of Internet Miscreants," in *Proc. of Conference on Computer and Communications Security (CCS)*, 2007, pp. 375–388.

[2] T. Holz, M. Engelberth, and F. Freiling, "Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones," in *Proc. of European Symposium on Research in Computer Security (ESORICS)*, 2009.

[3] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns," in *Proc. of USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[4] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the Storm and Nugache Trojans: P2P is here," *USENIX ;login:*, vol. 32, no. 6, pp. 18–27, 2007.

[5] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm." in *Proc. of USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[6] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," in *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2008.

[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," in *Proc. of USENIX Security Symposium*, 2008.

[8] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in *Proc. of European Symposium on Research in Computer Security (ESORICS)*, 2009, pp. 232–249.

[9] K. Rieck, G. Schwenk, T. Limmer, T. Holz, and P. Laskov, "Botzilla: Detecting the "phoning home" of malicious software," in *Proc. of 25th ACM Symposium on Applied Computing (SAC)*, March 2010, pp. 1978–1984.

[10] K. Borders and A. Prakash, "Web tap: detecting covert web traffic," in *Proc. of Conference on Computer and Communications Security (CCS)*, 2004, pp. 110–120.

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), Jun. 1999, updated by RFC 2817. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[12] D. Tax and R. Duin, "Support vector domain description," *Pattern Recognition Letters*, vol. 20, no. 11–13, pp. 1191–1199, 1999.

[13] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[14] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Neural Networks*, vol. 12, no. 2, pp. 181–201, May 2001.

[15] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *JMLR Workshop and Conference Proceedings, Volume 9: AISTATS*, Y. W. Teh and M. Titterington, Eds. MIT Press, 2010, pp. 405–412.

[16] ——, "Security analysis of online centroid anomaly detection," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-22, Feb 2010, coRR abs/1003.0078. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-22.html

[17] R. Beran and M. Srivastava, "Bootstrap tests and confidence regions for functions of a covariance matrix," *Annals of Statistics*, vol. 13, no. 1, pp. 95–115, 1985.

[18] G. Cretu, A. Stavrou, M. Locasto, S. Stolfo, and A. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *Proc. of IEEE Symposium on Security and Privacy*, 2008.

[19] M. Engelberth, F. Freiling, J. Goebel, C. Gorecki, T. Holz, R. Hund, P. Trinius, and C. Willems, "The InMAS approach," in *Proc. of European Workshop on Internet Early Warning and Network Intelligence (EWNI)*, Jan. 2010.

[20] C. Gorecki, "Truman box: Safe malware analysis by simulating the internet," Master's thesis, University of Mannheim, 2008.

[21] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2466, Dec. 1999.

[22] J. Beale, A. Baker, B. Caswell, and M. Poor, *Snort 2.1 Intrusion Detection*, 2nd ed. Syngress Publishing, 2004.

[23] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Proc. of IEEE Symposium on Security and Privacy*, 2005, pp. 120–132.

[24] W. L. R. Perdisci and N. Feamster, "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010, pp. 391–404.

[25] M. B. Salem and S. J. Stolfo, "Decoy document deployment for effective masquerade attack detection," in *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2011.

[26] V. Kemerlis, V. Pappas, G. Portokalidis, and A. Keromytis, "ileak: A lightweight system for detecting inadvertent information leaks," in *Proc. of European Conference on Computer Network Defense (EC2ND)*, 2011.

[27] J. McHugh, *Handbook for the Computer Security Certification of Trusted Systems*. Naval Research Laboratory, 1995, ch. Covert Channel Analysis.

[28] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. R. Karger, "Infranet: Circumventing web censorship and surveillance," in *Proc. of USENIX Security Symposium*, 2002, pp. 247–262.

[29] Y. Zhang and V. Paxson, "Detecting backdoors," in *Proc. of USENIX Security Symposium*, 200, pp. 157–170.