Security Monitoring of HTTP Traffic Using Extended Flows

Martin Husák, Petr Velan, Jan Vykopal
Institute of Computer Science
Masaryk University
Brno, Czech Republic
E-mail: {husakm, velan, vykopal}@ics.muni.cz

***Abstract*—In this paper, we present an analysis of HTTP traffic in a large-scale environment which uses network flow monitoring extended by parsing HTTP requests. In contrast to previously published analyses, we were the first to classify patterns of HTTP traffic which are relevant to network security. We described three classes of HTTP traffic which contain brute-force password attacks, connections to proxies, HTTP scanners, and web crawlers. Using the classification, we were able to detect up to 16 previously undetectable brute-force password attacks and 19 HTTP scans per day in our campus network. The activity of proxy servers and web crawlers was also observed. Symptoms of these attacks may be detected by other methods based on traditional flow monitoring, but detection using the analysis of HTTP requests is more straightforward. We, thus, confirm the added value of extended flow monitoring in comparison to the traditional method.**

I. Introduction

HTTP is currently the most widely used protocol which takes up a significant portion of network traffic. Due to its popularity, it is beneficial to have a deeper understanding of HTTP network traffic and its content. From a network security perspective, we would like to know who is accessing our network and what the requested resources are. Patterns in network traffic and outstanding numbers of visited hosts and requested resources would help us to distinguish between legitimate and malicious traffic.

The most suitable way of gaining an overview of HTTP traffic in a large-scale network is extended network flow monitoring [1]. There are two approaches to network traffic monitoring, deep packet inspection (DPI) and flow monitoring. DPI is resource demanding, but provides detailed information about a whole packet including a payload. Network flow monitoring is fast, but limited to Layers 3 and 4 of ISO/OSI model. Extended flow monitoring is a method combining benefits of both methods. It provides application-level data to traditional flow records while keeping the ability to monitor large-scale and high-speed networks. The correlation of logs from web servers is also an option, but in large networks it is not always possible to gain access to logs or even be aware of all of them. Therefore, our work is relevant for administrators of large networks in general, from academic networks to networks of ISPs.

We address two problems in this paper. The first one is the lack of an overview of network traffic and insufficient security awareness. This applies especially in heterogeneous networks with distributed administration. Many administrators oversee web servers in their administration and may oversee their neighbourhood, but they are not aware of security threats in the rest of the network. The second problem is to find a suitable set of tools to analyse HTTP traffic and distinguish between legitimate and malicious traffic. We are focused on large-scale network monitoring, where traditional flow monitoring approach cannot process application-layer traffic, while other approaches such as DPI cannot process large amount of data.

To formalize the scope of our work, we pose two research questions which we shall answer:

(i) What classes of HTTP traffic relevant to security can be observed at network level and what is their impact on attack detection?

(ii) What is the added value of extended flow compared to traditional flow monitoring from a security point of view?

The first question is focused on common types of HTTP traffic that we can observe in the network. We expect to extend the traditional division of network hosts into clients and servers. We focus on the behaviour of hosts regardless of their client/server role: we are particularly interested in detecting security-related patterns produced by attackers, proxies, and crawlers. We search for implications of the presence of particular traffic patterns. We aim to improve malicious traffic detection, filter out false positives, apply better security policy, and set up trustworthy honeypots.

The second question is focused on evaluating the contribution of extended monitoring with respect to the detection of malicious behaviour in the network. We focus on the detection of patterns in the network traffic that are hard to observe using standard flow. Extended flow would make detecting these events straightforward.

Our work contributes to a better understanding of current security threats. For example, we can detect vulnerability scanners and learn about vulnerability itself at the same time.

The answers to both questions are based on an analysis of real traffic in a campus network. We use network flow monitoring (IPFIX), extended by HTTP monitoring, which provides additional data to common network flow monitoring records. Although it is possible to extract some information, such as the Server Name Indication, from the HTTPS protocol, this work focuses on the HTTP protocol only.

We also use auxiliary methods to verify and validate the results, i.e., to confirm the membership of detected IP addresses into the proposed behaviour classes. Legitimate sources of traffic should be identifiable, e.g., Googlebot can be identified by a reverse DNS record of the source IP address [2]. We check User-Agents in HTTP traffic to differentiate legitimate and illegitimate traffic, e.g., requests with forged User-Agent. Finally, we check the reachability of traffic destination through search engines to see if it is possible to find the resource without accessing it. We follow the principle of "Google Hacking" [3], a technique used for finding vulnerabilities on web pages indexed by search engines.

This paper is divided into six sections. Section II presents previous work in this field. Measurement tools and environment are described in Section III. The results are presented in Section IV and discussed in Section V. Finally, Section VI concludes the paper.

II. Related work

Although research has focused on analysis of HTTP traffic in recent years, there is only a small number of papers relevant to our work. To the best of our knowledge, we are not aware of any paper dealing with HTTP analysis using a flow-based approach. In addition to this, the motivation of the relevant papers is different, typically focusing on efficient traffic management, with only a minor focus on security, specifically attack detection and prevention. One exception is the work by Perdisci et al. [4]. They employed network-level behavioural clustering of HTTP requests generated by malware. Their motivation was to provide quality input for algorithms that automatically generate network signatures. The second exception is the work by Husák and Čegan [5] in which they use flow-based HTTP analysis to detect users who contacted malicious websites.

Other related papers dealing with HTTP traffic analysis are listed in this paragraph. Augustin and Mellouk [6] published a classification of web applications according to three traffic features: intensity (total volume of exchanged traffic), symmetry (ratio between upstream and downstream traffic) and shape (burstiness over time). Xie et al. [7] addressed reconstructing web surfing behaviour from network packet traces. Xu et al. [8] analysed User-Agents strings captured at the edge of a campus network using an extension of UASparser [9]. They identified operating systems, types of devices (mobile, desktop) and applications (browser, crawler, P2P, automated updates and requests). A similar work by Jin and Choi [10] is motivated by efficient traffic management. Hur and Kim [11] proposed a smart phone traffic classification based on grouping User-Agent fields and extracting common strings.

III. Measurement tools and environment

This section describes the methods and tools used for acquiring the data from our campus network. We shall also provide characteristics of the network, in terms of its size and utilization.

We performed all measurements on the campus network of Masaryk University. The network has more than 40,000 users and 15,000 active IP addresses each day in /16 network segment. The network contains both servers and client stations, including proxy servers and a segment of honeypots. Any incoming traffic to the honeypots is by nature suspicious [12], which helps us to recognize malicious network traffic, e.g., HTTP requests on a honeypot web server. Only moderate filtering rules are applied globally to preserve the network neutrality of the academic environment.

Our primary source of data for network traffic analysis are FlowMon [13] probes located throughout the campus network. These probes use NetFlow and IPFIX protocols to export measured traffic as flow records to central collectors [1]. The collected flows are processed by various anomaly and intrusion detection systems, which report incidents to our CSIRT team. This network-centric approach allows us to monitor a complex network with many servers and services managed by different entities as opposed to host-based approaches such as log analysis. In such a case, it would be very difficult to even obtain system logs from every web server in such a varied environment. The monitored links are mostly 1 Gb/s and 10 Gb/s which makes the use of DPI systems impractical, or even impossible in the case of 10 Gb/s links, due to high processing requirements.

To capture data for this paper, we used a probe measuring traffic from a 10 Gb/s link which connects the campus to the ISP. We deployed extended measurement of the HTTP traffic on this probe using a flow exporter. The exporter software supports the extraction of basic elements from HTTP headers, such as host name, document URL (split into hostname and path), User-Agent string, and response code. These elements are then exported using the IPFIX protocol as enterprise elements [14]. We used an IPFIXcol [15] flow collector to receive and store the extended flow records.

A standard flow record consists of key elements (1) and of additional elements providing detailed information about the flow (2). The most common key elements are L3/L4 protocol numbers, IP addresses, and ports. Additional elements can contain any information from packet headers together with statistical counters, e.g., timestamps, number of packets and bytes, and autonomous system numbers.

$$F_{key} = (L3Proto, srcIP, dstIP, L4Proto, srcPort, dstPort) \quad (1)$$

$$F_{additional} = (timeStart, timeEnd, packets, octets, TCPflags, ToS, srcAS, dstAS) \quad (2)$$

A standard flow is then a concatenation of F_{key} and $F_{additional}$. The HTTP measurement [16] adds information from the HTTP application layer (3).

$$F_{HTTP} = (hostname, path, userAgent, requestMethod, responseCode, referrer, contentType) \quad (3)$$

The concatenation of a standard flow with application elements is called an extended flow. In this case, it will contain HTTP elements (4).

$$F_{ext} = F_{key} \cdot F_{additional} \cdot F_{HTTP} \tag{4}$$

We measured the F_{ext} vector in the environment of our campus network for three weeks over the summer break and three weeks in the semester. Thus, we could compare the amount of overall network traffic and suspicious events over two distinct time periods and traffic loads of the network. The amount of the traffic is lower in the summer break, although still comparable to the rest of the year. However, we assume that the amount of malicious traffic is constant over the year and is more apparent in the summer break.

Table I shows the size of each data set in packets, bytes, and flows. Moreover, it contains the number of HTTP requests observed. Although the number of requests is not high compared to the number of flows, there are at least as many responses, which often carry multimedia content. Therefore, the portion of HTTP traffic in bytes is very significant, despite the lower number of requests.

TABLE I
DATA SETS

Data set	Flows	Packets	Bytes	HTTP Req.
Summer 2014	3.733 G	188.953 G	198.362 TB	0.310 G
Spring 2015	6.680 G	552.523 G	604.704 TB	0.720 G

IV. RESULTS

Four characteristics of network flows were monitored: source IP address, destination IP address, hostname, and HTTP request. We analysed only HTTP traffic incoming to our network. The destination IP addresses were restricted to the /16 IP range of Masaryk University, while the source IP addresses were restricted to all other IP ranges. In some selected cases, we evaluated also the HTTP requests in opposite direction, i.e., source IP was in our network range and the destination IP was not. We used the hostnames and destination IP addresses interchangeably during the analysis due to a negligible amount of multi-hosting in our network. Overall, we observed HTTP traffic sent to almost 500 web servers, i.e., approximately 3 % of all hosts in our network. IPv6 traffic and measurement artifacts were observed in the measurement, but were omitted from the analysis due to their negligible impact on results.

The network traffic was analysed over five-minute and one-day intervals. A five-minute interval is the default granularity of data capture in network flow monitoring, i.e., one output file contains records of five minutes of network traffic. It allowed us to observe local anomalies, not only anomalies in the overall sample traffic. The one-day interval allowed us to compare anomalies over different days and discover repeated events and typical patterns. The one-day time window was chosen as it is long enough to contain an anomaly, while it is short enough in respect to the processing time of an analysis. The statistics presented in this section are based on an analysis of one-day intervals. However, for practical reasons, the five-minute interval is more convenient for detecting malicious events. We checked that the observed events can be detected even in the shorter time window.

We classified the network traffic according to the three main parameters of an extended flow: guest (source IP address), host (destination IP address or hostname), and HTTP request. We were interested in the cardinality of the relation between the parameters rather than values of the parameters. A high occurrence of flows sharing one or more parameters would suggest unusual activity in the network and is key for the classification. The number of flows which share the three main parameters is, therefore, an additional parameter for classification. The parameters and selected classes are presented in Table II.

TABLE II
TRAFFIC CLASSES

	#Guests	#Host	#HTTP Requests	#Flows
Class I	1	1	1	n
Class II	1	n	1	n
Class III	1	>1	n	>n

We were particularly interested in the results of three queries. First, how many similar requests were demanded by one source IP to one destination IP? Second, how many destination IP addresses were contacted by one source IP address with the same request? And third, how many requests were demanded by one source IP to one destination IP? Each query is represented by a class, as shown in Table II. The classes correspond to patterns in the network traffic as depicted in Fig. 1. A detailed description of each query and class can be found in the following subsections.

There are many other possible classes, although we found them to not be relevant from a security perspective. For example, many guests accessing a single host with a request suggests the popularity of a resource at the host. Such classes, however, could be interesting for general traffic classification, not security analysis.

A. Class I: Repeated requests

The first query asks for repeated requests between one guest and one host. We measured the number of occurrences of each triple, consisting of source IP, destination IP, and requested URL. Our assumption was that the repetition itself is common, but outstanding numbers may point to undesired behaviour. For example, password-protected web services may be exposed to brute-force attacks, which we can observe as a series of similar requests. The repeated request (A) is defined as follows:

$$\begin{aligned} RepeatedRequest(A) \iff A = \{F \mid \forall F, F' : \\ F(srcip) = F'(srcip) \wedge F(dstip) = F'(dstip) \\ \wedge F(path) = F'(path)\} \text{ and } |A| > threshold \end{aligned}$$

All flows in the set share the same source IP address, destination IP address, and requested HTTP path. The total number of flows in the set is bigger than the threshold.

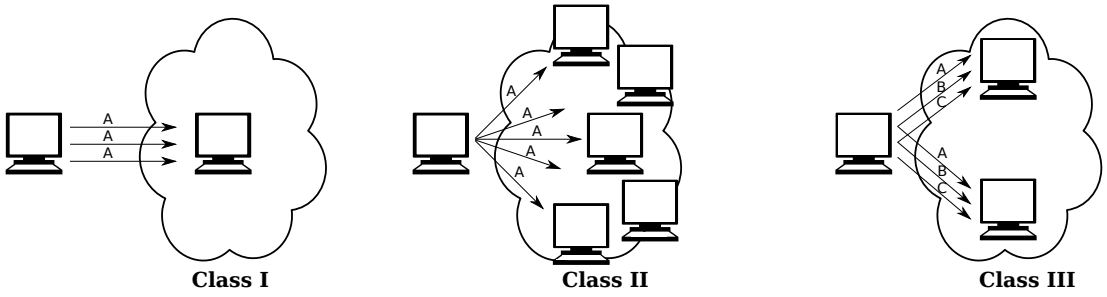

Fig. 1. Selected classes of HTTP traffic

Sample results are presented in Table III from one day in summer 2014. As we can see, the numbers of repeated flows reaches tens of thousands, which represents outstanding traffic patterns. There is a disproportion in the numbers of flows with the same source IP, destination IP, and HTTP request. While the majority of the triples were unique or repeated several times, a small number of triples were observed to be repeated several thousand times or more. The nature of these repeated requests was revealed by analysing the request. The majority of repeated requests contained a substring such as *admin* or *login*, which suggests that these traffic patterns were caused by brute-force attacks against password-protected web services. Other interesting repeated requests contained the substring *proxy*, which suggests communication between a client and a web-based proxy server. Furthermore, we found other repeated requests accessed large downloadable files, e.g., ISO images of Linux distributions.

TABLE III
TOP REPEATED REQUESTS FROM ONE DAY (DATA SET SUMMER 2014)

Guest	Host	HTTP Path	#Flows
G1	H1	/wp-**login**.php	46,031
G2	H2	/**admin**istrator/index.php	27,965
G3	H2	/**admin**istrator/index.php	27,798
G4	H3	/wp-**login**.php	25,316
G5	H4	/pub/linux/slax/Slax-7.x/7.0.8/slax-Chinese-Simplified-7.0.8-i486.iso	5,921
G6	H5	/**proxy**/lib**proxy**.pac	5,036
G7	H6	/node/	4,286
G8	H4	/pub/linux/slax/Slax-7.x/7.0.8/slax-English-US-7.0.8-i486.zip	4,170
G9	H7	/wp-**login**.php	3,632
G10	H7	/polit/wp-**login**.php	3,632

The second sample of results presented in Table IV was observed in one day in spring 2015. As we can see, there is a significant difference in the number of flows. However, the distribution of various types of repeated request is similar. One interesting anomaly is the guest G3, which performed repeated requests on 8 different hosts. This guest used different requests on different hosts, however the number of flows on each host is similar.

As we can see in Table V, which displays statistics for the whole monitored time window, almost half of the detected events were related to using a proxy. Brute-force password attacks were recognized only in 10.6 % of events. On the other hand, brute-force password attacks may generate more flows than other events in this class. They were the only events which generated more than 10,000 flows with the same source IP, destination IP, and HTTP path. However, we have also observed brute-force attacks of approximately a few thousand flows, but targeting multiple hosts over a short period of time.

TABLE IV
TOP REPEATED REQUESTS FROM ONE DAY (DATA SET SPRING 2015)

Guest	Host	HTTP Path	#Flows
G1	H1	/**proxy**/lib**proxy**.pac	5,147
G2	H2	/pub/linux/fedora/epel/6/x86_64/repodata/e63d28ff5a765b11a7052496f481f31aebab17c28545908d54e65904a1046ec8-filelists.sqlite.bz2	4,244
G3	H3	/senat/studenti/wp-**login**.php	3,992
G3	H4	/**admin**istrator/index.php	3,945
G3	H5	/slovnik/**admin**istrator/index.php	3,934
G3	H6	/**admin**istrator/index.php	3,926
G3	H7	/capv2011/**admin**istrator/index.php	3,924
G3	H8	/index.php	3,921
G3	H9	/**admin**istrator/index.php	3,794
G3	H10	/wp-**login**.php	3,701

TABLE V
DISTRIBUTION OF REPEATED REQUEST

Subclass	Path regular expression	Portion [%]
Proxy		**49.4**
	.*libproxy.pac	45.0
	.*sviproxy.pac	4.3
	.*proxy.php	0.1
Brute-force		**10.6**
	.*admin.*	6.7
	.*login.*	3.9
Others		**40.0**

B. Class II: Similar requests on many hosts

The second query addresses identical requests from one source IP address to many destination IP addresses. Our assumption was that a guest accesses only a limited number of hosts or at least requests different paths from different hosts. Therefore, a guest accessing a large number of hosts with the same request (with the exception of "/") is suspicious and worth noting. We would also like to point out the similarity between this traffic pattern and network port scanning, e.g., TCP SYN scan [17]. Therefore, we propose the name HTTP

scan for this traffic class. HTTP scan (B) is defined as follows:

$$HTTPScan(B) \iff B = \{F \mid \forall F, F' : F(srcip) = F'(srcip) \wedge F(dstip) \neq F'(dstip) \wedge F(path) = F'(path)\} \text{ and } |B| > threshold$$

All flows in the set share the same source IP address and requested HTTP path, while there are no flows with the same destination IP address. The total number of flows in the set is bigger than threshold.

First, we analysed only the scans of our monitored network by external hosts. Sample results from one day in summer 2014 are displayed in Table VI. For each scan we state a guest, i.e., the scanner, the requested HTTP path, the number of visited hosts, and the percentage of visited hosts of all hosts in the monitored network. There were two outstanding source IP addresses identified, one accessed almost all the web servers in our network and requested six paths. The second scanner accessed significantly less hosts with only one requested path. The other combinations of Source IP address and path, including the "/" request, were not observed to access more than ten hosts a day. We observed up to nineteen events a day in the measurement, including events where a guest requested more than one path. The same paths were also observed to be requested by different guests.

TABLE VI
TOP HTTP SCANNERS FROM ONE DAY (DATA SET SUMMER 2014)

Guest	HTTP Path	#Hosts	%
G1	/myadmin/scripts/setup.php	497	100
G1	/pma/scripts/setup.php	497	100
G1	/w00tw00t.at.blackhats.romanian.anti-sec:)	497	100
G1	/phpmyadmin/scripts/setup.php	495	99
G1	/phpMyAdmin/scripts/setup.php	494	99
G1	/MyAdmin/scripts/setup.php	491	99
G2	/manager/html	118	24

The number of detected HTTP scans varied from 3 to 19 per day over the monitored time window. Ten scans per day were detected on average and scanning for multiple paths was common. The scan for six paths, displayed in Table VI, was observed from four different IP addresses in one week. The other multiple-path scans did not use more than three paths in one scan.

The second sample results are from one day in spring 2015. We can see an interesting similarity of request in the first and second classes. The guest G1 was scanning for HTTP paths which also appeared in the results of the previous class. This suggests that the first class includes brute-force attacks and the second class some sort of network reconnaissance. Therefore, we may have observed the two phases of a complex attack.

Second, we searched for the HTTP scans in the opposite direction, i.e., scanning external networks by a host from our network. The detection in this direction is tricky as there are no clear boundaries of scanned networks and it is hard to set an appropriate threshold. In addition to this, even a common user may generate hundreds of similar requests such as */favicon.ico*. Therefore, we do not present any significant results. However, it would be possible to identify a scan by searching for continuous scanned network segments or the longest common prefix of the scanned IP addresses to estimate a scan of a subnet.

TABLE VII
TOP HTTP SCANNERS FROM ONE DAY (DATA SET SPRING 2015)

Guest	HTTP Path	#Hosts	%
G1	/wordpress/wp-login.php	337	68
G1	/site/wp-login.php	335	67
G1	/wp-login.php	332	67
G1	/blog/wp-login.php	201	40
G2	/bins.php	183	37
G3	/cgi-bin/test-cgi	102	21

C. Class III: Varying multiple requests on multiple hosts

The third query addressed the guests which requested a large number of unique paths on a single host. In this case, we seek guests which requested the majority of content from our hosts. Our assumption is that a large number of different requests from one guest to one host is a legitimate traffic pattern. On the other hand, outstanding numbers worth noting can also be observed. Then we looked for guests which requested an outstanding number of unique paths on more than one host. We assume that only crawlers behave in this way and it is unusual that any other guest would request such a large number of unique URLs on more hosts. The activity of a crawler (C) is defined as follows:

$$CrawlerCandidate(C') \iff C' = \{F \mid \forall F, F' : F(srcip) = F'(srcip) \wedge F(dstip) = F'(dstip) \wedge F(path) \neq F'(path)\} \text{ and } |C'| > threshold_1$$

$$Crawler(C) \iff C = \{F \mid \forall C'_i, C'_j : F_i \in C'_i \wedge F_j \in C'_j \wedge F_i(srcip) = F_j(srcip)\} \text{ and } |C| > threshold_2$$

We detect crawlers as guests which requested many resources from more hosts. First, all flows which have the same source IP address and destination IP address, but distinct requested HTTP paths, are grouped to sets. The first threshold is a minimal number of distinct requests on a single host. Second, we select source IP addresses which appeared in more sets. The second threshold is a minimal number of crawled hosts. The thresholds were selected to include at most 10 % of sets.

In the first phase, we confirmed that the assumed traffic pattern occurs frequently with a varying number of requested URLs. We were not able to unambiguously mark outstanding numbers. In the second phase, we filtered out the guests which did not access more than one host. The remaining guests were characterized by common signs, e.g., they accessed similar sets of hosts and requested a similar number of URLs. We have marked these guests as crawlers and present sample results from a one-day measurement in Tables VIII and IX.

Out of 11 crawlers detected in one day in summer 2014, 4 were identified as a well-known MSN search bot. The others were mostly local search engines, i.e., the Czech search engine

TABLE VIII
TOP HTTP CRAWLERS FROM ONE DAY (DATA SET SUMMER 2014)

Guest	Domain Name	#Hosts
207.46.13.62	msnbot-207-46-13-62.search.msn.com	7
157.55.39.107	msnbot-157-55-39-107.search.msn.com	6
137.110.244.137	bnserver2.sdsc.edu	4
157.55.39.156	msnbot-157-55-39-6.search.msn.com	4
157.55.39.6	msnbot-157-55-39-156.search.msn.com	4
37.187.28.19	z3.sentione.com	4
137.110.244.139	integromedb-crawler.integromedb.org	3
5.135.154.106	nks02.sentione.com	3
5.135.154.98	nks03.sentione.com	3
77.75.73.32	fulltextrobot-77-75-73-32.seznam.cz	3
77.75.77.17	fulltextrobot-77-75-77-17.seznam.cz	3

Seznam (2 guests) and social media monitoring tool Sentione (3 guests). The two remaining guests were identified as a part of an IntegromeDB crawler collecting biomedical data.

The second sample results are based on a one-day measurement in spring 2015. In the course of this day, we detected a higher activity of crawlers in our network. The recognized crawlers accessed significantly more hosts and generated more unique HTTP requests. We mostly observed well-known crawlers, such as the MSN search bot and two local search engines. One interesting finding was the detection of an unknown guest behaving like a crawler. The guest's IP address had no reverse domain name assigned and its WHOIS record pointed to a hosting company, which indicates a potentially malicious crawler.

TABLE IX
TOP HTTP CRAWLERS FROM ONE DAY (DATA SET SPRING 2015)

Guest	Domain Name	#Hosts
157.55.39.97	msnbot-157-55-39-97.search.msn.com	24
207.46.13.11	msnbot-207-46-13-11.search.msn.com	23
157.55.39.28	msnbot-157-55-39-28.search.msn.com	18
157.55.39.209	msnbot-157-55-39-209.search.msn.com	17
157.55.39.41	msnbot-157-55-39-41.search.msn.com	15
195.113.155.3	severus.mzk.cz	11
77.75.77.123	screenshotgenerator-77-75-77-123.seznam.cz	11
77.75.77.200	screenshotgenerator-77-75-77-200.seznam.cz	11
46.229.164.99	NOT FOUND	10

Similarly to HTTP scans, we were looking for the described activity in the opposite direction, i.e., crawling of external web servers by a crawler from our network. Again, the detection in the opposite direction is harder as there no clear thresholds and many potential false positives. The thresholds for crawler detection had to be set considerably higher to avoid false positives. However, no significant results demonstrating the advantages of this method were found.

V. DISCUSSION

In previous section we outlined three relevant classes of network traffic. The classes cover similar traffic patterns and can be characterized by outstanding numbers of observed network flows. The first class revealed several interesting patterns with ambiguous results as it can be split into several subclasses according to the HTTP request used. The second class is assumed to be solely malicious as it contains scanning for application vulnerabilities. The third class is considered mostly legitimate and is assumed to cover the activity of search bots and crawlers. However, not all the bots and crawlers are welcome in the network. Impacts on network security and the confirmation of the assumptions are discussed in this section for each class. There is also an interesting correlation between the first and second class which suggests that the malicious activity may be observed in both classes.

A. Class I: Brute-forcing and proxy servers

The first class is characterized by a large number of the same HTTP request from one guest to one host. As we can see in the results, several types of network activity are covered by this class. Generally, we cannot distinguish between them by the number of observed flows. The subclasses are easily distinguishable by analysing the requested path.

The first subclass was observed most often and was identified as communication between client and proxy. This subclass can be easily recognized by the substring *proxy* in a requested path. All the detected hosts were legitimate proxy servers in our network. The clients were located in networks of ISPs in our country, so we suppose the clients were legitimate. The implication for network security monitoring in this case is the possibility of proxy detection. We are able to detect active proxy servers in the network and identify their clients. Therefore, we should be able to reveal illegitimate proxy servers in the network or the illegitimate use of a proxy by unauthorized clients, e.g., from networks with a bad reputation [18].

The second subclass was not observed often, but generated the record number of flows. Paths containing a substring such as *admin* or *login* suggest a request to a resource protected by authentication. Paths ending in *wp-login* indicate the presence of an administrative interface of a well-known content management system WordPress, which is prone to brute-force password attacks [19]. This path was observed very often and the high number of flows, typically over a short period of time, indicates an attack. We accessed the requested URLs and confirmed that the hosts are running WordPress and provide a login page at the requested path. Another well-known content management system, Joomla!, was also a target of brute-force password attacks. The login page of Joomla! is located under a generic-looking path */administrator/index.php* and we have confirmed the presence of Joomla! at the observed URLs.

An interesting conclusion is that the dictionary attacks are rarely accompanied by any other network traffic from the source IP address. There were neither scans nor any other access to the host preceding the attack as is common among SSH brute-force attacks [20]. We suppose this is due to attackers using a "Google Hacking" technique [3]. In this technique, a search engine (e.g. Google, hence the name) is abused to do the reconnaissance for the attacker. The attacker searches a string typical for a WordPress login page and the search engine returns a list of WordPress instances which can be accessed instantly. Several well-aimed Google searches proved this assumption.

Although we can easily identify the two subclasses by the presence of characteristic substrings, the group of requested path is uncategorized and creates a third subclass. Paths such as */node/* and */wiki/* do not indicate a vulnerable resource. Dynamically generated content was observed on the requested URLs which suggests that the requests were legitimate. Highly repeated requests for URLs of files for downloading can be explained by partitioning the download, e.g., by download managers, and, thus, is legitimate traffic.

B. Class II: HTTP scanners

The second class was marked as HTTP scanning. Requested paths suggest that the scanning guests are searching for specific web applications and their vulnerabilities. It is also common that more paths, versions, or applications are probed during one scan. This type of traffic is easily detectable and highly interesting from a network security perspective. We have not observed any traffic that could be mistaken for a scan or marked as a false positive in this class. Even the low threshold of scanned hosts is sufficient for detection of a scanner as not every scanner accesses every host in the network. Fairly good results can be achieved with a threshold set to one fifth of the number of web servers in the network.

The HTTP scans were further analysed and correlated to TCP SYN scans on port 80. The results confirm the malicious nature of the scans. 46 % of the HTTP scans were preceded or accompanied by a TCP SYN scan of the full range of our network. The first option is that the scanner first obtains the list of IP addresses with port 80 opened and then scans them with HTTP requests. The delay between the two phases varies from one hour to several days. The second option of HTTP scanning is to send a HTTP request instantly after receiving a response to the TCP SYN packet.

TCP SYN scanning is easily detectable using network flow monitoring [21], but scanners can avoid detection by scanning "low & slow" [22]. We propose using extended flow monitoring to increase the detection rate of network scanning and lower the amount of false positives. We observed scanners that scanned no more than 5,000 IP addresses out of a /16 network range with a TCP SYN packet and continued HTTP scanning the web servers they found. Naturally, the scanners did not scan all the web servers, they typically found 100–250 hosts. On the other hand, as we stated earlier, more than 100 scanned hosts is enough to identify a scanner, at least in a network containing around 500 web servers. Therefore, we are able to identify a scanner which would otherwise avoid detection due to the high threshold of the TCP SYN scan detection method.

Another interesting property of the HTTP scans is that some of the requested HTTP paths were observed as a target of brute-force attack as well. For example, the scanners were looking for running instances of WordPress by requesting paths ending in *wp-login*. The same paths were subjects of thousands of repeated request, which we consider as a brute-force attack. This lead us to suggestion that the attackers search for victim of a brute-force attack using the HTTP scans, just like it is common for attackers to perform port scan before a brute-force attack against services such as SSH [20].

C. Class III: Web crawlers

The third recognized class of HTTP traffic was marked as crawlers. The query had to be further specified due to the interchangeability of crawlers and common guests when measured on a single host. Covering more hosts in the network provided us with a set of crawlers active in our network. The crawlers were further analysed to filter out false positives. Filtering methods consisted of querying reverse DNS records, querying the WHOIS database, and checking User-Agents used in HTTP requests. We confirmed the identified set was populated by legitimate crawlers. The User-Agents in the HTTP headers pointed to well-known search engines such as Googlebot, Bing, and Baidu. Reverse DNS and WHOIS records further confirmed the presumptions by referring to domains names of the search engine owners.

Crawlers are mostly legitimate and welcome in the network [23]. Almost all the crawlers we discovered were confirmed as legitimate. However, there are two reasons why we included them in a security-related analysis. First reason is small number of IP addresses that were identified as crawlers using the flow-based method, but we were not able to tell which any further information about them. Missing reverse DNS records or empty User-Agent fields in HTTP queries lead to suggestion that these IP addresses performed illegitimate crawling, e.g., e-mail harvesting that aims at discovering spam recipients [24].

The second reason of including crawlers in the analysis is the large number of flows they generate. Any potential detection method based on extended flow analysis would have to deal with false positive alerts. Legitimate web crawlers can be easily mistaken for malicious traffic due to their omnipresence and large number of requests. On the other hand, we have to be careful about false negatives, i.e., malicious hosts disguising as crawlers to avoid attention. In addition to this, operators of web crawlers may change the addresses of their bots over time or not publish them at all, which makes creating a whitelist a difficult task. Our method is based on monitoring the behaviour of the potential crawler, which alleviates the possibility of false positives or false negative detections. The User-Agent field of HTTP extended flow record can also be used for validation of the results.

VI. Conclusion

We presented an analysis of HTTP traffic using extended flow monitoring in a large campus network. Contrary to previously published analyses, we were the first to focus on the security aspects. We identified three security-related traffic classes: repeated requests, HTTP scans, and web crawlers. Repeated request were further split into brute-force password attacks and proxies according to the observed HTTP request.

This classification is based on an analysis of selected elements of extended flow: source IP, destination IP, and

requested URL split into domain and path. Our analysis supported the hypothesis that flow monitoring extended to HTTP headers (Layer 7) enables the straightforward detection of current security issues compared to traditional flow monitoring performed at Layers 3 and 4. Particularly, the brute-force password attacks and proxies are hard to differentiate using only traditional flow monitoring.

Malicious traffic is contained in the classes of HTTP scans and repeated request, in which we were able to identify brute-force attacks against well-known content management systems. Web crawlers represent a class of legitimate traffic that can be mistaken for malicious activity due to its large number of generated flows. However, suspicious crawlers can also be observed. We have also identified the use of proxies as a traffic class. Using a proxy is not malicious by nature, but is interesting for security enforcement. For example, the presence of an unauthorized proxy may violate security policy in the network.

The classification is based on counting number of extended flows which share one or more parameters. The classification can be thus easily converted into a threshold-based detection method. We were able to detect malicious HTTP traffic in large-scale network and process it from one point with a low amount of false positives. Ten previously undetectable brute-force password attacks and 10 HTTP scans per day were detected on average in our campus network. In addition, previously unknown proxies and web crawlers were observed. The large-scale network monitoring approach proved beneficial particularly during the detection of HTTP scans and web crawlers, which are hardly detectable using only local system logs or local monitoring.

Further work will involve the integration of the proposed methods into the existing detection framework in the campus network. False positive and false negative ratios of the detection methods based on our proposed methods will be evaluated. The security incidents concerning HTTP will be resolvable globally in the whole network and the security awareness can be spread more effectively. Additionally, we will use the discovered classes for setting up more efficient honeypots to respond to the requested vulnerability. This applies particularly to HTTP scanners and brute-force password attacks. We should be able to equip a honeypot with an appropriate resource, e.g., vulnerable web application that the adversaries were looking for, to attract and analyse specific attacks. The honeypot use case involves both network monitoring and security incident analysis and can lead to an effective reaction to upcoming threats including a reaction to zero-day attacks.

REFERENCES

[1] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.

[2] "Verifying Googlebot." [Online]. Available: https://support.google.com/webmasters/answer/80553

[3] J. Billig, Y. Danilchenko, and C. E. Frank, "Evaluation of Google Hacking," in *Proceedings of the 5th Annual Conference on Information Security Curriculum Development*, ser. InfoSecCD '08. New York, NY, USA: ACM, 2008, pp. 27–32.

[4] R. Perdisci, W. Lee, and N. Feamster, "Behavioral Clustering of HTTP-based Malware and Signature Generation Using Malicious Network Traces," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010.

[5] M. Husák and J. Čegan, "PhiGARo: Automatic Phishing Detection and Incident Response Framework," in *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*, 2014, pp. 295–302.

[6] B. Augustin and A. Mellouk, "On Traffic Patterns of HTTP Applications," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec 2011.

[7] G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, and Y. Jin, "ReSurf: Reconstructing web-surfing activity from network traffic," in *IFIP Networking Conference, 2013*, May 2013.

[8] Y. Xu, G. Xiong, Y. Zhao, and L. Guo, "Toward Identifying and Understanding User-Agent Strings in HTTP Traffic," in *Web Technologies and Applications*, ser. Lecture Notes in Computer Science, W. Han, Z. Huang, C. Hu, H. Zhang, and L. Guo, Eds. Springer International Publishing, 2014, vol. 8710, pp. 177–187.

[9] J. Mallat, "UASparser." [Online]. Available: http://user-agent-string.info/download/UASparser

[10] C.-G. Jin and M.-J. Choi, "Integrated analysis method on HTTP traffic," in *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*, Sept 2012.

[11] M. Hur and M.-S. Kim, "Towards smart phone traffic classification," in *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*, Sept 2012.

[12] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, 1st ed. Addison-Wesley Professional, 2007.

[13] INVEA-TECH a.s., "FlowMon probe." [Online]. Available: https://www.invea.com/en/products-and-services/flowmon/flowmon-probes

[14] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011 (INTERNET STANDARD), Internet Engineering Task Force, Sept 2013.

[15] P. Velan and R. Krejčí, "Flow Information Storage Assessment Using IPFIXcol," in *Dependable Networks and Services*, ser. Lecture Notes in Computer Science, R. Sadre, J. Novotný, P. Čeleda, M. Waldburger, and B. Stiller, Eds., vol. 7279. Heidelberg: Springer Berlin Heidelberg, 2012, pp. 155–158.

[16] P. Velan, T. Jirsík, and P. Čeleda, "Design and Evaluation of HTTP Protocol Parsers for IPFIX Measurement," in *Advances in Communication Networking*. Springer, 2013, pp. 136–147.

[17] M. H. Bhuyan, D. Bhattacharyya, and J. Kalita, "Surveying Port Scans and Their Detection Methodologies," *Comput. J.*, vol. 54, no. 10, pp. 1565–1581, Oct. 2011.

[18] G. C. Moreira Moura, "Internet bad neighborhoods," Ph.D. dissertation, University of Twente, Enschede, March 2013. [Online]. Available: http://doc.utwente.nl/84507/

[19] WordPress Codex. (2014) Brute Force Attacks. [Online]. Available: http://codex.wordpress.org/Brute_Force_Attacks

[20] J. Vykopal, M. Drašar, and P. Winter, *Flow-based Brute-force Attack Detection*. Stuttgart: Fraunhofer Verlag, 2013, pp. 41–51.

[21] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 3, pp. 343–356, Third 2010.

[22] M. Kang, J. Caballero, and D. Song, "Distributed Evasive Scan Techniques and Countermeasures," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. Lecture Notes in Computer Science, B. M. Hämmerli and R. Sommer, Eds. Springer Berlin Heidelberg, 2007, vol. 4579, pp. 157–174.

[23] Y. Sun, I. Councill, and C. Giles, "The Ethicality of Web Crawlers," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, vol. 1, Aug 2010, pp. 668–675.

[24] O. Hohlfeld, T. Graf, and F. Ciucu, "Longtime Behavior of Harvesting Spam Bots," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 453–460.