

Horizon Extender: Long-term Preservation of Data Leakage Evidence in Web Traffic

David Gugelmann
ETH Zurich, Switzerland
gugelmann@tik.ee.ethz.ch

Dominik Schatzmann
ETH Zurich, Switzerland
schatzmann@tik.ee.ethz.ch

Vincent Lenders
armasuisse, Switzerland
vincent.lenders@armasuisse.ch

ABSTRACT

This paper presents Horizon Extender, a system for long-term preservation of data leakage evidence in enterprise networks. In contrast to classical network intrusion detection systems that keep only packet records of suspicious traffic (black-listing), Horizon Extender reduces the total size of captured network traces by filtering out all records that do not reveal potential evidence about leaked data (white-listing). Horizon Extender has been designed to exploit the inherent redundancy and adherence to protocol specification of general Web traffic. We show in a real-life network including more than 1000 active hosts that Horizon Extender is able to reduce the total HTTP volume by 99.8%, or the outgoing volume by 90.9% to 93.9%, while preserving sufficient evidence to recover retrospectively time, end point identity, and content of information leaked over the HTTP communication channel.

Categories and Subject Descriptors

C.2 [COMPUTER-COMMUNICATION NETWORKS]:
General—*Security and protection (e.g., firewalls)*

Keywords

network security, data leakage, network forensics, incident investigation, aggregation.

1. INTRODUCTION

Digital forensics deals with collecting and preserving evidence for post mortem analysis of security incidents. Network traffic traces, including full payload content and aggregated network flows, prove to be very valuable for the forensic investigation of network-related data theft. For this reason, enterprises and government organizations increasingly deploy network forensics systems that preserve traces of network activity in their own network infrastructure.

Preserving complete network trace records for extended periods of time is becoming almost impossible because of

ever increasing traffic volumes. That is why commercial network packet forensic collection systems focus their full payload collection on a limited time window, usually consisting of the last few days to weeks [1, 8]. However, since many real-world data theft incidents remain undetected for months or even years before being investigated, the relevant evidence is often missing. As an example, hackers have been suspected of having had deep access for nearly one decade to the IT systems of the former telecom giant Nortel Networks Corp. without being detected [10].

An existing approach to preserve network forensic evidence of security breaches for longer periods is to trigger packet collection when alarms are raised by a network intrusion detection system (NIDS). For example, *Time Machine* [14] or *Snort* [2] are systems that keep packet traces of selected flows that are identified as suspicious using predefined signatures. The drawback is that network evidence is only preserved for known threats, i.e., when signatures for the threats are known and activated a priori. However, it is well known that classical NIDS have significant false negatives rates, which in practice leads to missed relevant leakage evidence. Similarly, dynamically reducing storage space by packet sampling [15] leads to loss of evidence.

We introduce Horizon Extender, an alternative approach for reducing the total volume of collected traffic while preserving sufficient evidence of data leakage. Horizon Extender addresses the threat of a hacker or malicious insider extracting sensitive information from a network. He or she could try to steal data without being detected by hiding it in the noise of normal outbound traffic. Rather than searching for data leakage signatures at run-time, Horizon Extender aims to constrain the collected data to only records that may represent relevant evidence of leaked data.

Our work focuses specifically on HTTP traffic as this is the protocol accounting for a large portion of the overall traffic. Furthermore, many organizations, particularly those in which confidentiality is a high priority, will generally only allow outbound HTTP traffic, which is forced to go through a proxy server, and block everything else. Horizon Extender exploits the fact that HTTP traffic is highly redundant [3, 16] and mainly determined by the protocol specifications [5]. We exploit these two characteristics to reduce the total volume by filtering out forensically irrelevant HTTP fields and removing the redundancy of the remaining fields by means of deduplication. Yet, Horizon Extender preserves sufficient evidence to recover *time*, *host identities*, and *content* of a leakage incident, therefore covering three key parameters of interest in network forensics.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ASIA CCS'13, May 8–10, 2013, Hangzhou, China.

Copyright 2013 ACM 978-1-4503-1767-2/13/05 ...\$15.00.

We have implemented Horizon Extender and evaluated its performance on HTTP network traffic obtained from an academic campus network with more than 1000 hosts over the period of seven days. Our main contribution is to show that Horizon Extender, which uses a combination of filtering, deduplication, data ordering, and compression, allows the reduction of the total HTTP volume required for archival by 99.8%, or by 90.9% to 93.9% of the outbound traffic, while preserving enough evidence for the investigation of data leakage incidents. The time horizon, in which data leakage evidence is retained, is therefore 11 to 16 times longer for the same storage budget when using Horizon Extender than when using raw HTTP log data.

Related work on the compression of HTTP traffic [3, 16] mainly focused on removing the redundancy in HTTP flows originating from servers. This paper focuses in contrast on HTTP client requests, i.e., the outbound HTTP traffic of monitored workstations. While server responses usually contain a message body that is several kilobytes in size, the average number of bytes sent with every client request is on the order of some hundred bytes. Repeated strings in HTTP requests are quite short, therefore current deduplication approaches that operate with block sizes on the order of several kilobytes to reduce the overhead caused by deduplication [12] are not efficient for HTTP requests. Like [8], Horizon Extender relies on various levels of aggregation; however, Horizon Extender not only uses time but also white lists and Web site popularity for selecting data for aggregation. In contrast to behavior-based outgoing HTTP traffic anomaly detection [4], our work applies behavior-based classification for identifying traffic that will likely be irrelevant for retrospective data leakage investigations. Our work is targeted at efficient preservation of information leaks in Web traffic, whereas a method to quantify information leaks is proposed in [5].

2. SETUP AND FORENSIC USE CASES

Network Setup. We consider a typical enterprise network setup. An internal network comprising client PCs is connected to the Internet through a dedicated HTTP proxy. The proxy blocks all traffic except HTTP and HTTPS Web traffic, which is intercepted and replicated to Horizon Extender for archival. The proxy provides clear text access to TLS-secured HTTP traffic so that Horizon Extender can handle HTTPS similar to HTTP.

Problem Statement. Although storage and computational costs have dropped steeply, archiving and processing fine-grained information on every HTTP stream on the network for long periods is currently impractical. For example, a router in our academic network observes an average of 2.4 TB/day of HTTP traffic. If we were to record every HTTP packet for a year, that would require about 870 TB of storage. Trying to store and process this volume of traffic would be massively expensive. In practice however, only a small portion of the traffic turns out to be relevant for forensic analysis of data leakage incidents. Nevertheless, it is difficult to know a priori what data is relevant as every HTTP request line may potentially be exploited to leak data. The key challenge is therefore how to decide a priori what data will be crucial when subsequently investigating an incident retrospectively. An additional important requirement in forensics is determinism. Aggregation and filtering of network records must be deterministic and

reproducible over the entire processing chain. That is, it must be possible to tell whether a network record is missing because it did not happen or because the information was not retained. While deterministic processing is simple to achieve when archiving all network data in a first-in-first-out (FIFO) manner, this determinism is more challenging to guarantee for schemes that only record network traffic on behavior- and signature-based detection. Horizon Extender bridges the gap between traditional forensic tools that focus on determinism and completeness of the retrieved data [11] without concern for processing speed and storage space and anomaly detection based approaches [2, 13].

Attacker Model. We address the threat of a hacker or malicious insider extracting sensitive information from the internal network. He or she tries to steal data without being detected by hiding it in the noise of normal outbound HTTP traffic. For Web traffic, this often means stashing bytes in paths or header fields within seemingly benign requests. We assume that the attacker got root access on a set of client PCs within the internal network. The number of clients that the attacker may control is however limited to a maximum number (e.g., 10) or a fixed percentage (e.g., less than 10%) of the clients in the internal network. From these client PCs, he may therefore generate any traffic at any time. However, all leaked data must eventually pass through the HTTP proxy such that the attacker cannot escape the monitoring by Horizon Extender. Another assumption is that the attacker is not able to spoof IP addresses; this can be enforced by the IEEE 802.1X standard.

An attacker may further attempt to exploit hidden information channels. Since all connections go through the proxy server which terminates TCP connections, covert channels using fields in the TCP or IP header are not possible. However, an attacker may still establish a covert timing channel [7] by shaping the timing of requests, but the data leakage rate of such a channel is rather small. Since the problem of detecting and reducing the bandwidth of hidden information channels has already been studied [7], we will not discuss it further in the context of Horizon Extender. The scope of our work is therefore limited to attackers that leak information represented by legitimate content in the fields of the HTTP protocol. When the information is obfuscated (e.g. by encryption), Horizon Extender itself may not be able to decode the leaked content, but provides at the least an archived version of the obfuscated data.

Forensic Use Cases. The goal of Horizon Extender is to provide an administrator with the ability to analyze data leakage incidents retrospectively. Network forensic investigations are typically conducted on the *connection meta information* level or on the *payload* level. In particular, Horizon Extender focuses on the retention of network forensic evidence for the following generic purposes:

Attribution: Attribution deals with assigning leaked content, e.g., to attribute a known file that has been uploaded to a client and remote server identity.

Repudiation: Repudiation deals with providing assurance that hosts are innocent. That is to prove that particular hosts have not been involved in data leakage, e.g., by proving that a connection or upload did not take place.

Exploration: When a client has been identified as malicious by any external means, the collected evidence may serve to reconstruct the complete transfer history and to explore leakage of sensitive data.

3. TRAFFIC CLASSIFICATION

This section introduces the classification model on which Horizon Extender relies to classify HTTP sessions.

3.1 Trusted Service Model

Horizon Extender classifies traffic according to a trusted service model. The traffic is classified at the level of HTTP sessions. An HTTP session in HTTP/1.1 [9] is a sequence of network request-response transactions defined by methods like GET, POST, HEAD, etc. To decide which part of requests and responses should be archived, Horizon Extender differentiates two basic classes of sessions: sessions to *trusted* and *untrusted* Web services. Trusted Web services are trusted not to collude with malicious internal clients while untrusted services may potentially do so. When a service is trusted, the service is assumed to be well-behaving and the HTTP requests may only leak data over the reserved data fields of each request. For example, as shown in Figure 1, data leakage must happen over the request URI (line 1) or the message body (line 14) as reserved for uploading parameters. All other fields such as for example the CONNECTION or COOKIE fields are protocol specific and reserved for connection establishment and tracking. When the server is trusted and behaves according to the protocol specification, those fields do not pose a real data leakage threat, since they are internal connection state, only available to the hosting server, and not accessible to service users.

In contrast, a malicious service may collude with an internal client to leak data over any header fields of HTTP. It is therefore unsafe to assume that reserved fields will not be exploited by untrusted services for data leakage.

3.2 Popularity-assisted Classification

To classify HTTP sessions in *trusted* and *untrusted* services, Horizon Extender relies on a semi-automated behavior based classification scheme. Horizon Extender compiles a popularity ranking of accessed services by analyzing the percentage of internal hosts communicating with each service and displays the most popular services to the administrator for classification. This popularity-based classification has three major advantages: (1) the most popular services are likely to attract most traffic and therefore inherently cover a large portion of the total traffic volume; (2) popular Web services (e.g., Google, Facebook, etc) are more likely to be trustworthy¹; (3) malicious manipulation of the service popularity is not trivial for an internal attacker since the popularity ranking is not volume based, thus a manipulation of the popularity may trivially be prevented by requiring a minimum number of distinct clients to define a service as popular. From the list of popular services, the administrator checks the hostname² of the Web services and classifies the services as trusted or untrusted. For trusted services, the administrator further categorizes the service as *interactive* or *non-interactive*. This distinction will help to further reduce the storage space requirements by discarding irrelevant

¹The fact that popular services are trusted does not mean that the service users are also trusted. It just states that the server is trusted to behave according to the HTTP specification.

²Services are classified by hostname because Web servers in the same domain might have different properties, e.g., static files included in an interactive Web site might be retrieved from servers in a dedicated subdomain that points to a CDN.

```
1: POST /sec/wp-comments-post.php HTTP/1.1
2: Host: blogs.example.com
3: User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:7.0.1)
[...]
8: Connection: keep-alive
9: Referer: http://blogs.example.com/sec/read-the-classics/
10: Cookie: __utma=116342964.432951434.1334764398.
    1331764393.1337764332.1; __utmb=116042265.4.15.1321764092;
11: Content-Type: application/x-www-form-urlencoded
12: Content-Length: 229
13:
14: author=john&email=john%40mail.example.com&url=&comment=
    A+comment+in+a+blog+might+be+used+to+leak+data&submit=Post
```

Figure 1: Sample POST request (lines truncated)

content. Interactive services provide internal clients with the ability to download and upload content (e.g., Gmail, Facebook, Youtube, etc). Non-interactive services are pure download services where users retrieve content without being able to upload content (e.g., software updates hosted by CDNs or advertisement servers). While the HTTP upload data is relevant for interactive services, the session content of trusted non-interactive services is not relevant as these services do not pose a data leakage threat. For brevity, we will use the terms *trusted* for interactive trusted services and *white-listed* for non-interactive trusted services. This leads to the following three basic class definitions: *untrusted*, *trusted*, *white-listed*.

4. FORENSIC DATA STORAGE

At its core, Horizon Extender differentiates four aggregation levels for the representation of HTTP traffic records. The four aggregation levels are referred to as L1, L2, L3, and L4 with the higher the number the more aggregated are the network records contained in the aggregation level:

L1 (Raw HTTP): L1 represents a chronological raw trace of the complete HTTP requests as intercepted by the proxy. This data is further enriched with connection meta information including *time*, *source IP address*, *destination IP address*, *source port*, *destination port*, and *host*. *Host* is the hostname (DNS name) of the Web service. This level is used for short-term caching of incoming flows.

L2 (Compressed HTTP): At this aggregation level, L1 raw records are split into connection meta information records and reassembled data streams. The HTTP data streams are further split on newlines such that every line of outgoing HTTP traffic is transformed into one data block. Deduplication also operates on these data blocks, as explained in Sect. 5. This aggregation level is used for preserving flows classified as *untrusted*.

L3 (URI and message BODY): Level L3 uses the same meta information records and data structure for data blocks as level L2. To further save storage space, only data blocks containing request URIs and message bodies are retained. This aggregation level only captures data leakage that is conducted by submitting data over the message body or the request URI and is therefore used to store sessions classified as *trusted*.

L4 (Flow end points): In L4, only the meta information records of the HTTP flows are stored. This highly aggregated data representation allows the analysis of a client's remote communication end points and the amount of data that was sent and received. The HTTP content is however

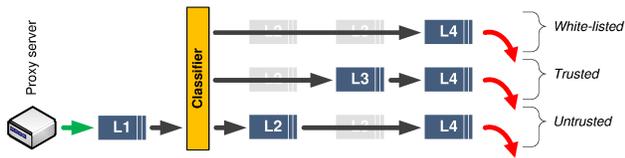


Figure 2: Data processing and archive states.

no longer available. This aggregation level is used to store the category of *white-listed* flows and old flows of the *trusted/untrusted* classes.

5. PROCESSING AND DEDUPLICATION

This section explains how Horizon Extender processes data being exported by the proxy server as visualized in Fig. 2.

Processing of Incoming Records. If approved traffic classification rules are already in place, the incoming L1 records are immediately classified and aggregated to the storage level assigned for the respective traffic category. After aggregation, records are deduplicated and cached until a time or size limit³ is reached. When the limit³ is reached, the cached meta information records and data streams are ordered by client IP and remote service, compressed using gzip, and written to two separate files.

Deduplication is applied at L2 and L3 to remove redundancy in the traffic. Deduplication reduces storage size by replacing repeated byte sequences with a pointer to the previous storage location. Instead of using general content-aware block boundary calculation approaches like rolling hashes or Rabin fingerprints [6], Horizon Extender exploits that different HTTP fields are separated by newline and treats every line of the HTTP stream as one data block. Thus, the line number of a previous occurrence can be used as a pointer. Since the main source of redundancy in HTTP requests are static header fields that are sent with every request, this simple approach is efficient, especially in combination with ordering data by client IP and service and applying a file compression algorithm to the resulting deduplicated file. Ordering data blocks by client IP address and service improves the overall compression performance because data blocks of connections to the same service or from the same client often have byte strings in common (e.g., URIs of images located in the same directory). Thus, similar byte strings are moved close to each other such that they fall within the compression window of gzip.

Aggregation of Archived Data. When the storage budget assigned to records in the L2 and L3 aggregation levels is exhausted, Horizon Extender aggregates the oldest L2 and L3 records to the aggregation level L4, which means that corresponding payload data blocks are deleted. When the storage budget for aggregated L4 records is reached, Horizon Extender deletes the oldest connection meta information entries. Horizon Extender guarantees deterministic processing by grouping incoming network records by time and always moving complete groups of network records through the different archival states. Therefore, a group of network records is either in a state in which all relevant payload content is available given that services have been classified correctly or in a state in which only connection meta information is available for the investigation of data leakage.

³one day in the current implementation

6. EVALUATION

6.1 Methodology

To evaluate Horizon Extender, we collected HTTP traces at the border router of a university network. A selected subnetwork is used for the evaluation, covering different departments of the university. Our trace spans seven full days including Monday through Wednesday during one week in late 2012 and Thursday through Sunday on the following week. Based on the IPv4 address, we identified per day between 671 and 1888 regular internal clients issuing HTTP requests. The corresponding HTTP traffic on TCP port 80 sums up to 22.6 GB of outgoing and 1.1 TB of incoming data. We additionally classified clients as *active* on a daily basis if they communicated with more than 50 different Web servers per day, these were 112 to 1093 clients per day. Only active clients have been used for calculating the popularity ranking of Web services.

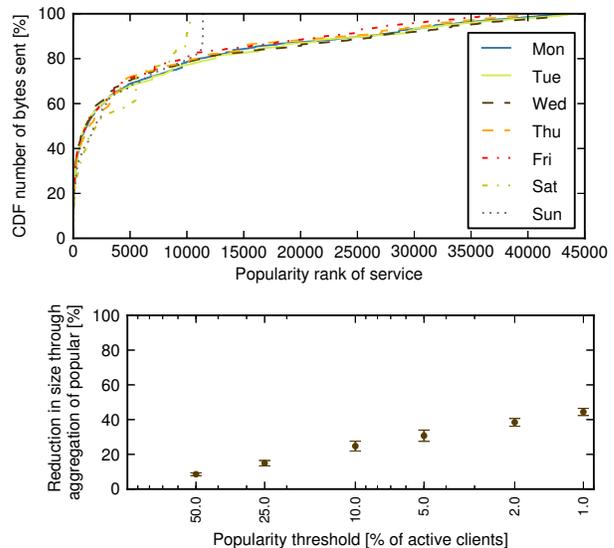


Figure 3: Bytes sent to HTTP services (top). Average reduction in size through aggregation of popular services from L2 to L3 (bottom).

6.2 Traffic Classification

Horizon Extender uses the popularity of Web services to compile traffic classification rules. In this section, we first discuss the correlation between popularity and traffic volume and show the impact of different popularity thresholds. Second, we discuss a list of services suggested to the administrator by Horizon Extender.

Analysis of Service Popularity. The cumulative distribution for the number of bytes sent to services vs. the popularity of services is shown in Fig. 3 (top) for every day. For the weekend days, there is a steep increase at the end of the curves. This is caused by individual clients uploading files to otherwise unpopular services.

Horizon Extender compiles a list of popular services by selecting all services whose popularity is higher than a certain threshold. The popularity is expressed as the percentage of active clients in the network visiting it. Fig. 3 (bottom) shows for different popularity thresholds by which percent-

age the average total volume of the data streams is reduced when aggregating corresponding connections from level L2 to L3. Because the popularity rank distribution (not shown) is heavy-tailed with the tail starting at around 10%, the volume savings are rising sharply once the popularity threshold is below 10%. Based on these numbers, we have set the popularity threshold for weekdays to 2% (around 20 clients).

Popular Service List. Next, we analyze how stable the list of popular services is. Fig. 4 shows how many new services appear every day on the list. We see that only few new services appear every day and that the number of new services is in general decreasing. Since the set of popular services is quite stable, Horizon Extender can use the same classification for multiple days to weeks with small degradation in compression performance. Table 1 presents the 15 most popular services for the *Mon* dataset in order to give the reader an intuition.

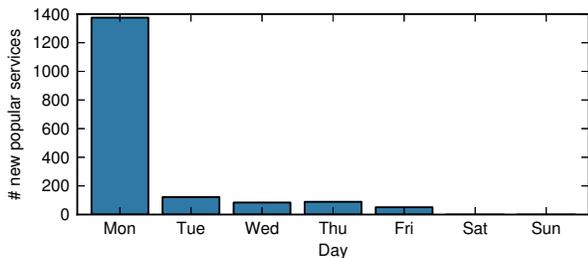


Figure 4: Number of new services appearing.

Host	Pop.	P1	P2	P3
www.google-analytics.com	87.7%	U	T	W
www.google.com	78.9%	U	T	T
ocsp.verisign.com	76.3%	U	T	W
www.facebook.com	75.8%	U	T	T
googleads.g.doubleclick.net	74.0%	U	T	W
www.google.ch	73.2%	U	T	T
platform.twitter.com	71.4%	U	T	W
s0.2mdn.net	70.6%	U	T	T
r.twimg.com	69.1%	U	T	T
p.twitter.com	68.9%	U	T	W
cdn.api.twitter.com	67.6%	U	T	T
connect.facebook.net	64.8%	U	T	W
pagead2.google syndication.com	64.4%	U	T	W
ocsp.thawte.com	64.1%	U	T	W
b.scorecardresearch.com	63.7%	U	T	W

Table 1: The 15 most popular services identified in the *Mon* dataset together with their classification according to different policies (U: untrusted, T: trusted, W: white-listed).

6.3 Effectiveness of Horizon Extender

Overall effectiveness. Next, we analyze the effectiveness of Horizon Extender for three administrative policies: **Policy P1 - no trust:** An administrator behaving according to this policy does not trust any services. Therefore all services are classified as *untrusted* (U).

Policy P2 - trust but no white list: The administrator closely follows Horizon Extenders suggestions and classifies

	Week days	Weekend	Overall
raw	100.0±0%	100.0±0%	100.0±0%
raw gzip	23.4±0.65%	30.0±2.17%	23.8±1.92%
P1 (L2)	8.4±0.73%	18.5±1.88%	9.1±2.75%
P2 (L2-L3)	5.8±0.79%	15.4±2.06%	6.5±2.63%
P3 (L2-L4)	5.4±0.79%	15.1±2.08%	6.1±2.66%
L4	0.3±0.01%	0.2±0.01%	0.3±0.01%

Table 2: Size of processed records compared to the size of raw records.

all services as *trusted* (T) that Horizon Extender identifies using its popularity classification algorithm.

Policy P3 - trust and white list: All suggestions are classified as *trusted*, additionally the administrator goes through the 100 most popular services and white-lists (W) services, for which he thinks that they are non-interactive.

The list of popular services derived from the *Mon* dataset has been reviewed according to the three policies (see Tab. 1 for an excerpt) and has then been used to classify and accordingly aggregate connections; L2-L4 records have been deduplicated, ordered and compressed (using gzip) as discussed in Sect. 5, further meta information of L2 and L3 records has been aggregated for the administrative policies P2 and P3.

The results are shown in Tab. 2. Overall, the average size is reduced to 9.1% - 6.1% of the original outgoing data size, which is a factor of 2.6 to 3.9 better than only gzipping the raw data. Applying policy P2 instead of P1 reduces the total volume by approximately 29% and using P3 squeezes out another 6%. The difference between P2 and P3 is quite small because we only considered the top 100 services for white-listing and we assumed that there are no fully-trusted and therefore white-listed Web sites (e.g. a Web-based business management system) to which users upload content. The compression rate is worse at the weekend because file uploads account for a higher share of egress HTTP requests. Since block boundaries are set on newlines, which is not optimal for binary data, and because a file that is only seen once does not bring much redundancy, such data streams cannot be well deduplicated and Horizon Extender’s compression ratio degrades towards the one of gzip.

In summary, Horizon Extender extends the time horizon for forensic investigations by a factor of 15 to 16 compared to raw log data and by a factor of 3.7 to 3.9 compared to compressed raw log data.

Analysis of Deduplication. Horizon Extender uses deduplication to efficiently store repeated data blocks. The impact of deduplication is illustrated in Fig. 5 for the *Mon* dataset. The upper plot shows the effect of deduplication on the total unique block count for selected request fields and the lower plot on total storage size required by the corresponding blocks. General header fields like **HOST** or **USER-AGENT** can be very well deduplicated. This is because they are contained in every request and usually stay constant for many requests. **COOKIE** blocks account for most volume. **GET URI** blocks account for most volume after deduplication since Web browsers usually cache requested objects and therefore do not send multiple requests for the same resource. In summary, deduplication reduces the total number of unique data blocks by 91.2% and the total file size or volume by 68.8%. Anand et al. reported a redundancy of 20% and 76% for

outgoing HTTP traffic in a campus and large company network, respectively [3]. Our findings are therefore in line with what they reported for the large company network.

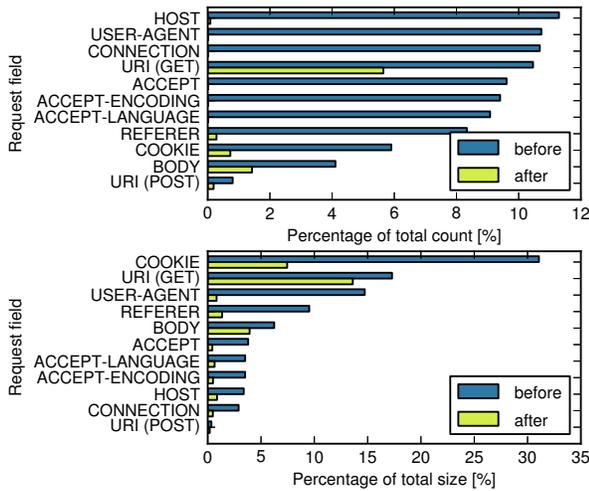


Figure 5: Impact of deduplication on total count (top) / total size (bottom).

7. DISCUSSION AND CONCLUSIONS

We have presented Horizon Extender, a network forensic collection system for HTTP traffic that aggregates records in contrast to traditional systems not merely according to their age but also according to their likelihood of being relevant for future security investigations. Horizon Extender relies on traffic classes to decide which parts of network records might be relevant for incident investigations. We have shown that the popularity of Web services is stable over several days to weeks and therefore efficient filters for traffic classification can be built with little manual configuration by analyzing connection patterns. By taking the different traffic classes into account and quickly aggregating and eventually discarding parts of network records that are not relevant for forensic investigations, Horizon Extender extends the time window in which information relevant for incident investigations is available by a factor of 15 to 16 in relation to the total outbound HTTP traffic volume.

Evasion in the context of Horizon Extender means that there is no evidence in Horizon Extender's logs for some data that has been leaked from a network. Covert timing channels have already been discussed in Sect. 2. Another attack is to deceive an administrator into marking a service as trusted even though the service is controlled by the attacker. However, assuming that the administrator updates the list of popular hosts only once a month, attackers must control multiple internal clients for more than a month, otherwise they cannot push the service to the list of popular hosts and there is still no guarantee that the administrator marks the service as trusted. Another attack would be to take over the servers of a popular service that has already been marked as trusted. However, since popular services generally are operated by large companies that monitor their services closely, this remains a challenging task.

8. ACKNOWLEDGEMENT

This work was partially supported by the Zurich Information Security Center. It represents the views of the authors.

9. REFERENCES

- [1] Netwitness - Investigator. <http://netwitness.com/>. Last visited: 2012-12-03.
- [2] Snort - Network intrusion prevention and detection system. <http://www.snort.org/>. Last visited: 2012-12-03.
- [3] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee. Redundancy in network traffic: findings and implications. In *Proc. of ACM SIGMETRICS '09*, 2009.
- [4] K. Borders and A. Prakash. Web tap: detecting covert web traffic. In *Proc. of ACM CCS '04*, 2004.
- [5] K. Borders and A. Prakash. Quantifying information leaks in outbound web traffic. In *Proc. of IEEE Symp. on Security and Privacy*, 2009.
- [6] A. Broder. Some applications of Rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*, pages 143–152. Springer Verlag, 1993.
- [7] S. Cabuk, C. E. Brodley, and C. Shields. Ip covert timing channels: design and detection. In *Proc. of ACM CCS '04*, 2004.
- [8] E. Cooke, A. Myrick, D. Rusek, and F. Jahanian. Resource-aware multi-format network security data storage. In *Proc. of ACM SIGCOMM workshop on LSAD '06*, 2006.
- [9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, 1999.
- [10] Financial Post. Nortel computers breached by hackers for over a decade: WSJ. <http://business.financialpost.com/2012/02/14/nortel-computers-breached-by-hackers-for-over-a-decade-wsj/>. Last visited: 2012-12-03.
- [11] S. L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7:S64–S73, Aug. 2010.
- [12] F. Guo and P. Efstathopoulos. Building a high-performance deduplication system. In *Proc. of 2011 USENIX*, 2011.
- [13] S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer. Building a time machine for efficient recording and retrieval of high-volume network traffic. In *Proc. of ACM SIGCOMM IMC '05*, 2005.
- [14] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider. Enriching network security analysis with time travel. *SIGCOMM Comput. Commun. Rev.*, 38(4):183–194, Aug. 2008.
- [15] A. Papadogiannakis, M. Polychronakis, and E. Markatos. RRDtrace: Long-term Raw Network Traffic Recording using Fixed-size Storage. In *2010 IEEE MASCOTS*, 2010.
- [16] N. T. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. *ACM SIGCOMM Comput. Commun. Rev.*, 30(4):87–95, 2000.