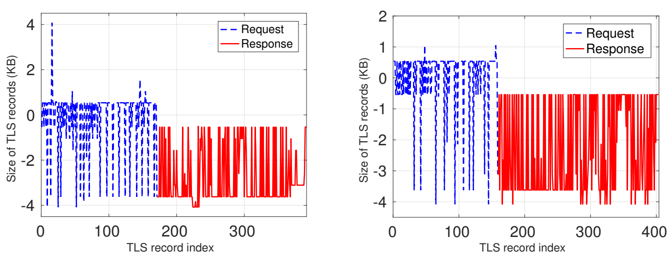
Poster: Fingerprinting Past the Front Page: Identifying Keywords in Search Queries over Tor

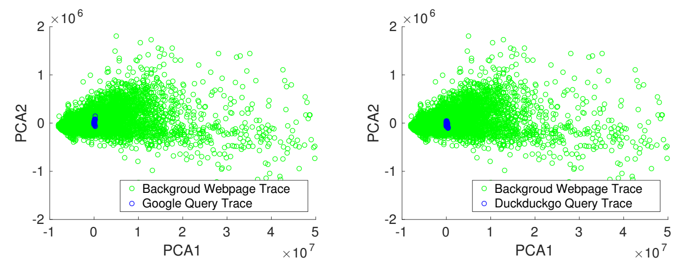
Se Eun Oh
University of Minnesota
seoh@umn.edu

Nicholas Hopper
University of Minnesota
hopper@cs.umn.edu



(a) Keyword pizza (b) Keyword target

Fig. 1. TLS records in the Google query trace. (+) indicates outgoing packets and (-) indicates incoming packets



(a) Google (b) Duckduckgo

Fig. 2. Principal Component Analysis (PCA) Plot of Google and Duckduckgo query traces and background webpage traces based on CUMUL feature set

Search queries a user makes to Internet Search Engines contain a great deal of private and personal information about the user. Thus, popular search engines such as Google and Bing, and ISPs, are in a position to collect sensitive details about users. These search queries have also been among the targets of censorship [7] and surveillance infrastructures [2] built through the cooperation of state and private entities. One of mitigations against such privacy leaks is to use Tor [10], where the identity of clients is concealed from servers and the contents and destinations of connections are concealed from network adversaries, by sending connections through a series of encrypted relays. However, Tor cannot always guarantee the user anonymity since the timing and volume of traffic still reveal some information about the user browsing activity, which has been actively explored in Website Fingerprinting (WF) researches. [1], [3], [5], [8], [9], [11], [12]

In this work, we describe a new type of traffic analysis attack on Tor, a Keyword Fingerprinting (KF). In this attack model, a local passive adversary attempts to infer keywords that users query, based only on analysing traffic intercepted between the client and the entry guard in the Tor network. A KF attack proceeds in two stages. First, the attacker must identify which Tor connections carry the search result traces of a particular search engine against the other webpage traces. The second is to determine whether a target query trace is in a list of "monitored keywords" targeted for identification or to classify each query trace correctly to predict the keyword that the victim typed.

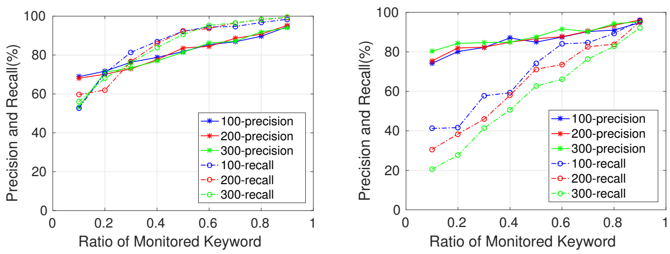
In particular, we discover new task-specific feature sets focusing on the specific portion of the search query trace, called "response" portion (Figure 1), and demonstrate the feasibility of the KF attacks using Support Vector Machine (SVM) [4] with a variety of experiment settings. As shown in Figure 2, existing feature sets used in WF do not carry sufficient information for identifying specific keywords, we conduct an in-depth feature analysis using Kruskal-Wallis H test [6]. Based on χ^2 statistics, we selectively choose feature sets where each keyword group has statistical difference enough to be identified by the KF and aggregate them, named Aggr4 in this work.

TABLE I
TPR, FPR, and within-monitored accuracy comparing to those of cumulTLS [8].

Metric	cumulTLS	Aggr4
TPR(%)	34.95	82.56
FPR(%)	3.94	8.09
WM-Accuracy(%)	0.01	56.52

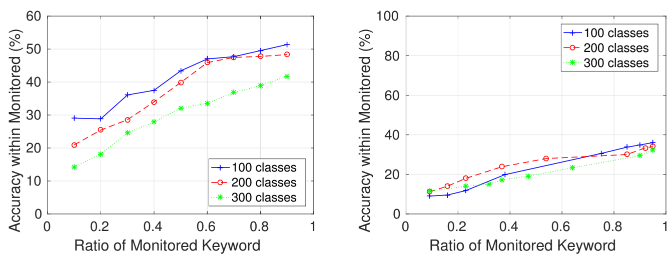
Table I presents that while state-of-the-art WF features [8] perform very well for the first stage of our attack, our feature sets, Aggr4, significantly improves the accuracy in the second stage identifying keywords. This new feature set is powerful across different experiment settings.

As shown in Figure 3, when we vary the size of monitored and background keyword sets, both metrics decrease with increasing the size of background set, however the size of monitored set has no impact on those in the binary-label learning and minimal impact on the precision in the multi-label learning. Based on Figure 4, when we consider different Tor Browser settings, the incremental search setting with



(a) Binary-Label Learning (b) Multi-Label Learning

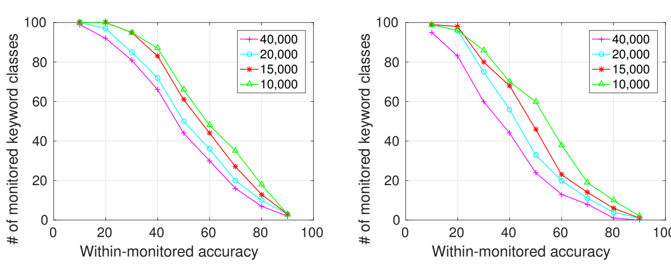
Fig. 3. Precision & recall for binary classification when varying the number of monitored and background Google keywords (Note that ratio means |monitored set|:|total set| and we used 30 instances for each monitored keyword)



(a) Incremental Query Setting (b) One Shot Query Setting

Fig. 4. Within-monitored (WM) accuracy for multi-class classification when varying the size of classes and instances of monitored Google keywords (Note that we used 30 instances for each monitored keyword)

Java Script (JS) enabled (by default) such as Google Instant ensures better WM accuracy (The number of traces from monitored keywords classified with the correct label over the total number of monitored traces.) than "high security" search with JS disabled (via Noscript configuration). This is because the former carries additional rich information such as traffic for auto-complete. According to Figure 5, the KF can be applicable to most search engines since their query responses contain an informative response portion, however the degree of fingerprintability varies with the search engine. Google shows better WM accuracy because it discloses additional traffic pattern led by incremental search results returned by Google



(a) Google (b) Bing

Fig. 5. Within-monitored (WM) accuracy CCDF when varying the size of classes of background keywords (Note that we use 80 instances for each 100 monitored keyword and 10k∼40k background keywords)

Instant. The binary classification further makes it feasible to identify "related keyword" searches containing keywords that are not in the training set but are semantically closed to monitored keywords (TPR=68.8% and FPR=0.0005% to detect 20,000 related searches). Finally, we investigate the relationship between the degree of complexity in search result HTML and the fingerprintability of that keyword, which helps to understand how search engines and users mitigate such attacks.

In conclusion, all experimental results indicate that use of Tor alone may be inadequate to defend the content of users' search engine queries.

Note that while KF and WF attacks share some common techniques, the KF focuses on the second stage of this attack, distinguishing between multiple results coming from a single web application, which is challenging for existing WF techniques. As shown in Table I, when differentiating between monitored keywords, classifiers based on recent WF features perform no better than random guessing (0.01%). Thus, the different level of application as well as the multi-stage nature of the attack make it difficult to directly use or compare results from the WF setting.

REFERENCES

[1] X. Cai, X. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," *Proceeding of the 2012 ACM conference on Computer and Communications Security*, pp. 605–616, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2382260

[2] G. Greenwald and E. MacAskill, "NSA Prism Program Taps in to User Data of Apple, Google and Others," *The Guardian*, June 2013. [Online]. Available: http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data

[3] D. Herrmann, R. Wendolsky, and H. Federrath, "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier," *CCSW*, 2009.

[4] C. Hsu, C. Chang, and C. Lin, "A Practical Guide to Support Vector Classification," *BJU international*, vol. 101, pp. 1396–400, 2008. [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/papers/guide/guide.pdf

[5] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A Critical Evaluation of Website Fingerprinting Attacks," *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, pp. 263–274, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2660267.2660368

[6] W. H. Kruskal and W. A. Wallis, "Use of Ranks in One-Criterion Variance Analysis," *Source Journal of the American Statistical Association*, vol. 4710087, pp. 583–621, 1952. [Online]. Available: http://www.jstor.org/stable/2280779$\backslash$nhttp://www.jstor.org/page/

[7] J. Ng, *Blocked on Weibo: What Gets Suppressed on China's Version of Twitter (And Why)*. New Press, The, 2013.

[8] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website Fingerprinting at Internet Scale," *16th NDSS (NDSS 16)*, pp. 143–157, 2016.

[9] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," *WPES*, 2011. [Online]. Available: http://lorre.uni.lu/{∼}andriy/

[10] T. Project, "Tor homepage," https://www.torproject.org/, Tor Project.

[11] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting," *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 143–157, 2014.

[12] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society - WPES '13*, pp. 201–212, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2517840.2517851

Fingerprinting Past the Front Page: Identifying Keywords in Search Queries over Tor

Se Eun Oh Nicholas Hopper
University of Minnesota

Abstract

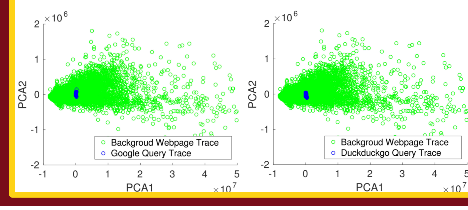
- In this work, we introduce a Keyword Fingerprinting (KF), extending Website Fingerprinting (WF), to identify keywords in search queries. Based on a two-stage, traffic analysis-based approach with new task-specific feature sets, a passive network adversary can defeat the use of Tor.
- We demonstrate the feasibility of the KF attacks across four popular search engines and various experimental settings (e.g., user query setting). We also further explore why several keywords are better fingerprintable.

Keyword Fingerprinting (KF)

- The attacker will progress through two sequential fingerprinting steps.
  - 1st step: Webpage fingerprinting to identify the query result traffic of the specific search engine
  - 2nd step: KF to predict keywords in query traces by both binary and multi-class classification
- KF focuses on 2nd step, which is challenging for existing WF techniques.

KF vs. WF

- CUMUL classifiers proposed by Panchenko et al. perform very well for the 1st step, which detects blue against green area. However, when identifying and differentiating keywords in blue, classifiers based on WF features perform poorly.



RESP feature set

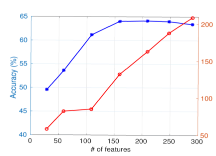
- All 80,000 query traces included a long sequence of incoming packets at the end of the trace. We call it "Resp" and remaining portion "Request".
- **Resp** is more informative than the request portion

Metric	Google		DuckDuckgo	
	RQ	RP	RQ	RP
Avg of # of packets	140	223	102	193
Max # of packets	288	559	251	801
Avg of total payload(KB)	115	496	89	434
Max of total payload(KB)	350	1246	295	1669
SVM Accuracy(%)	13.88	17.22	14.69	20.83

- We extracted **Resp** feature sets; Total number of TLS records, max, mean, sum of TLS record sizes (**RespTotal**); Sequence of cumulated size of TLS records (**cumulRespTLS**); Sequence of the corresponding number of Tor cells (**cumulRespTorCell**)

Data Preparation

- Reverse cumulRespTLS and cumulRespTorCell
  - The last elements are total size of TLS records and total number of Tor cells in Resp and good features to identify search terms
  - SVM accuracy for the first and last 140 packets in cumulRespTLS: 21.33% vs. 53.79%
- Number of Features: Use 247 features as it gave the best accuracy as well as acceptable running time



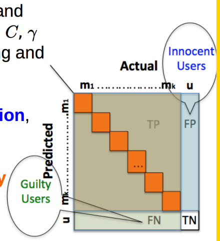
Feature evaluation using χ^2 statistics

- We tested different combinations of feature sets whose χ^2 statistics was higher than 6,000 and the best feature set was "**Aggr4**" aggregating Total, RespTotal, RcumulRespTLS, and RcumulRespTorCell

Feature	SS	MS	χ^2
roundedTCP	4.5e+10	4.55e+8	1353
roundedTLS	6.35e+10	6.42e+8	1905
cumulTLS	7.08e+10	7.15e+8	2123
Total	2.15e+11	2.17e+9	6461
burstIncoming	2.8e+11	2.83e+9	8402
RcumulRespTLS	2.22e+11	2.24e+9	6667
RcumulRespTorCell	2.17e+11	2.19e+9	6528

Support Vector Machine

- We used a non-linear classifier with a radial basis function (RBF) and 10-fold cross validation to find C, γ and to split dataset into training and testing set.
- Metrics
  - Binary Classification: **Precision**, **Recall** (TPR), FPR (%)
  - Multi-class classification: **Within-monitored Accuracy** (WM-acc) (%)



Confusion matrix for multiple outcomes

TPR and FPR when we identify 10k Google and Duckduckgo query traces against 100k webpage traces

- Google query trace identification

Ratio	0.1	0.2	0.3	0.5	0.8
TPR(%)	99.82	99.82	99.95	99.84	99.84
FPR(%)	0	0	0.0001	0.0001	0
precision(%)	100	100	99.98	99.99	100

- Duckduckgo query trace identification

Ratio	0.1	0.2	0.3	0.5	0.8
TPR(%)	99.94	99.94	99.96	99.94	99.94
FPR(%)	0	0	0	0	0
precision(%)	100	100	100	100	100

**Ratio means Monitored set size : Total set size

Closed and Open World Experiment

- Closed-world accuracy (10k keywords and 100 classes)

feature	Accuracy(%)
Total	35.48
torCell	7.54
roundedTCP	12.73
roundedTLS	15.16
burstIncoming	26.7
cumulTLS	18.67
RespTotal	26.14
RespTLS	17.22
RcumulRespTorCell	53.43
RcumulRespTLS	53.79
Aggr2	62.23
Aggr3	63.43
Aggr4	64.03

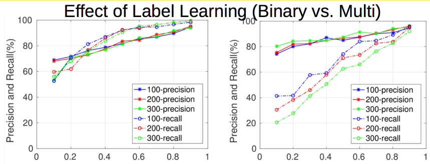
- Identifying 100 monitored keywords against 10k background keywords

Metric	Binary-label	Multi-label
TPR(%)	93.12	82.56
FPR(%)	14.88	8.09
Precision(%)	86.27	91.11

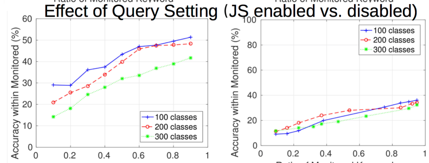
- Comparison to CUMUL classifier

Metric	cumulTLS	Aggr4
TPR(%)	34.95	82.56
FPR(%)	3.94	8.09
WM-Accuracy(%)	0.01	56.52

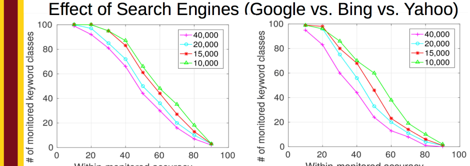
Effect of Label Learning (Binary vs. Multi)

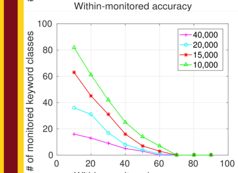


Effect of Query Setting (JS enabled vs. disabled)



Effect of Search Engines (Google vs. Bing vs. Yahoo)





Identifying 20k related searches using Binary classification

Metric	80ins
TPR	68.8
FPR	0.0005
Precision	99.85

TPR and Analysis on search result HTML

TPR(%)		# link	# domain	# Tag	# attribute
Google	0	49	10	845	1,575
	40	72	11	1,014	1,989
	80	84	14	1,378	2,749
Bing	0	33	10	406	533
	40	42	12	461	654
	80	118	18	826	1,410
Yahoo	0	46	1	527	928
	40	106	1	820	1211
	80	N/A	N/A	N/A	N/A

TPR(%)		max depth	# block	# tag direction change	len(HTML)	len(Data)
Google	0	24	37	244	128k	1,684
	40	30	49	319	165k	2,030
	80	35	62	449	232k	2,745
Bing	0	13	32	142	44k	807
	40	12	41	170	47k	914
	80	14	77	378	58k	1,635
Yahoo	0	18	30	191	92k	1,048
	40	20	63	390	96k	1,638
	80	N/A	N/A	N/A	N/A	N/A

** block=count # Blocks based on depth=18 for Google, 9 for Bing, and 14 for Yahoo, len()=number of characters