Packet Content Anonymization by Hiding Words

José Zamora Ponce
Dpto. Ingeniería Matemática
Universidad de Chile
jzamora@dim.uchile.cl

Martin Loebl
Department of Applied Mathematics
Charles University
Prague, Czech Republic
loebl@kam.ms.mff.cuni.cz

Lukas Kencl
Intel Research
15 JJ Thomson Avenue
Cambridge, CB3 0FD, UK
lukas.kencl@intel.com

***Abstract*— This demo shows a novel technique to anonymize payload data contained in network traces, while preserving the capability of verifying the presence of short substrings of certain length. The method is based on shuffling the substrings of the input payload. It can be shown that it is very hard to reconstruct the original payload, while presence of a particular short string can be verified with a low false positive rate (and no false negatives). This enables network traces payloads to be made available for searches for malicious content (e.g. worms), while preventing reconstruction of the actual data in the payload.**

I. Introduction

It is a hard problem for network researchers and network operators to store, and make available for studies, networking traces containing entire packets with their payload portions [1]. Such task is difficult due to the concerns of possibly revealing sensitive information, either of private or business nature. However, such ability would also be very desirable, as testing or investigating many networking algorithms nowadays requires access to the packet payload. In particular, in the domain of network security and intrusion detection systems (IDS), many methods work (e.g. the Snort [2] IDS or the Autograph [3] Worm detector) operate by performing a deep packet inspection, matching the packet (or reassembled stream) against a multi-pattern database of malicious sequences.

In this work, we have postulated the following goal: designing an anonymizing technique that would make impossible to interpret and reconstruct the content of the payload, yet still enable search throughout the anonymized content for e.g. malicious keywords of certain maximal length.

The presented algorithm is loosely based on card shuffling, as analyzed in [4].

II. Mathematical Model

A. Problem Statement

We introduce a more precise model for the above *hiding-word* problem. The *hiding-word* problem is the following: Given a word ω and a set S of forbidden words, we want to transform ω to another word ω_F so that:

I.- The size of ω_F is at most for $c|\omega|$, where c is a constant.

II.- Let k be the size of the longest word in S. Then, if s is a subword of ω with $|s| \leq k$, then s is a subword of ω_F.

III.- With low probability a forbiden word not in ω appears in ω_F.

IV.- It is HARD to reconstruct the word ω from ω_F

B. Shuffling Algorithm

Let A be a finite alphabet. A word ω of size n is an element of A^n. We denote by $\omega_i \in A$ the i coordinate of ω. We denote by $\omega(i,j)$ the subword of ω that starts in the coordinate i and finishes in the coordinate j. Let s_1, s_2 be two words then $s = s_1 \cdot s_2$ will denote their concatenation.

Let ω be a word in A^n and m, k integers with $n > km$. We define the following operation that we call (k,m)-*shuffle* of a word ω:

1) We choose a random sequence of integers $n_0, n_1, n_2, \ldots, n_m$ with $n_0 = 0$, $n_m = n$ and $n_{i+1} \geq k + n_i$ for $i = 0, \ldots, m-1$.
2) For $i = 1, \ldots, m$ we define the card $s_i = \omega(n_{i-1}+1, n_i)$.
3) Let $s'_1 = s_1$ and for $i = 2, \ldots, m$ $s'_i = s_{i-1}(n_{i-1}-k+1, n_{i-1}) \cdot s_i$.
4) We radomly choose a permutation π in S_m, the symmetric group. We create the word $\omega_f = s'_{\pi(1)} \cdot s'_{\pi(2)} \cdot \cdots \cdot s'_{\pi(m)}$.

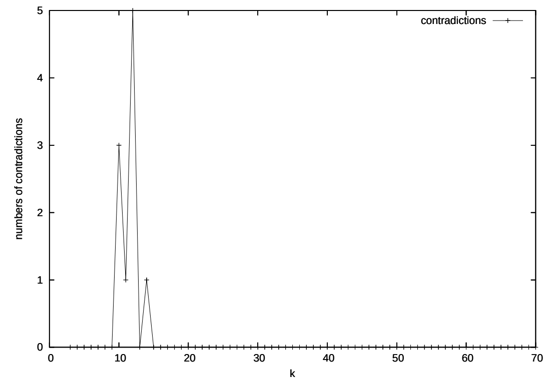

Fig. 1. False positive rate on a boolean alphabet, with input string of length $n = 1500$ characters, cut into $m = 15$ substrings to be shuffled.

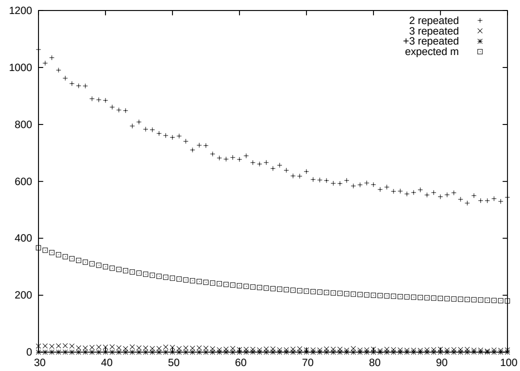

Fig. 2. Number of substring repetitions on a boolean alphabet, with input string of length $n = 12000$ characters, cut into $m = 100$ substrings to be shuffled. On the horizontal scale is k, the size of the forbidden word. The m_F (square symbol) curve denotes the expected number of cards of the two steps of the algorithm, while other curves show the number of repetitions.

For $k < s_1 < s_2$ a variant of the previous operation is a (k, s_1, s_2)-*shuffle* , where the size of the card is chosen randomly between s_1 and s_2. The value of m is random in this case. Hence we choose a sequence of integers $n_0, n_1, n_2, \ldots, n_{m'}$ with $n_0 = 0$, $n_1 = r_1$ and while $n_i < n - k$, $n_{i+1} = n_i + r_i$ with r_i a random number between s_1 and s_2. Finally $n_{m'} = n$.

Note that if s is a subword of ω whose size is less than k, then s is a subword of ω_f always.

Shuffling Algorithm: Now we are ready to propose our hiding algorithm, which we call (k, m)-*supershuffle*. It consists of the following two steps:

- We $(k, k, 2k)$-shuffle ω and we obtain ω_f.
- We (k, m)-shuffle ω_f and we obtain the word ω_F.

The final word ω_F is our proposal for hiding of ω.

We note that:

$$|\omega_f| = n + \frac{2n}{3k}(k-1) \leq 2|\omega| \ ;$$

$$|\omega_F| \leq |\omega_f| + m(k-1) \leq 3|\omega|.$$

Hence the model carries out the conditions I and II of the anonymization problem. In the following, we discuss the conditions III an IV respectively.

III. Preliminary Experimental Results

A. False Positives

We want to estimate the probability that a subword s that is not in ω appears in ω_F. We call this event a *false positive* or a *contradiction*.

We use a 2-letter alphabet ($A = \{0, 1\}$) for the experiments. We create a random word ω, we run the algorithm and then we create a random word s of size k that is not in ω and we check if it is in ω_F.

Figure 1 suggests that the probability of a false positive is very low. Indeed, in the case when the input is larger, the probability tends to zero.

B. Reconstruction

The key question is how hard is it to reconstruct the word ω if we know ω_F. The crucial information for the attacker is the repetition of words of certain size. If the number of repetitions is equal to the number of the cards then it is easy to reconstruct the original word.

We conduct experiments to calculate the numbers of repetitions of words of size k restricted to the $\{0, 1\}$ boolean alphabet. Note that the expected total number of cards is $m_F = \frac{2|\omega|}{3k} + m$. Figure 2 looks optimistic for non-constructibility. For different values of the parameters, the results are very similar as in Figure 2, meaning that generally there are no more than 3 repetitions of a word.

In real world, we usually work with data expressed in Bytes. Hence we conducted experiments where every character is represented by an 8-bit digit, this is $|A| = 2^8$, and we examine the sizes for forbidden words of 4 and 8 Bytes, substring sizes shown to be useful for quick malicious string search in [5]. The results of studies on texts of size close to a typical packet size of 1500 Bytes can be seen in tables I and II. While again quite optimistic, it is important to note that performance is somewhat worse on real text, due to inherent repetitions in these.

	k=4	k=8
2 repetitions	326	211
3 repetitions	22	10
+3 repetitions	0,5	0,2
m_F	350	225

TABLE I

NUMBER OF REPETITIONS, 8-BIT ALPHABET, n=1500, m=100, RANDOM GENERATED INPUT.

	k=4	k=8
2 repetitions	185	258
3 repetitions	91	33
+3 repetitions	154	9
m_F	483	341

TABLE II

NUMBER OF REPETITIONS, 8-BIT ALPHABET, n=1702, m=200, REAL TEXT INPUT.

IV. CONCLUSION

The shuffling method seems to be a good candidate for anonymizing packet payload, while preserving searchability of short substrings. Perhaps as its greatest drawback can be perceived the growth in size of the packet (up to three times). However, we believe that a clever compression scheme might be able to reduce the size back to the original, due to redundancies created by the shuffling.

REFERENCES

[1] R. Pang and V. Paxson, "A high-level programming environment for packet trace anonymization and transformation," in *Proceedings of ACM SIGCOMM*, 2003.

[2] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proceedings of LISA '99: 13th Systems Administration Conference*, Seattle, WA, November 1999.

[3] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection," in *Proceedings of the 13th Usenix Security Symposium (Security 2004)*, San Diego, CA, August 2004.

[4] B. Mann, "How many times should you shuffle a deck of cards?" in *UMAP Journal, Vol. 15, Pages 303-332*, 1994.

[5] R. Ramaswamy, L. Kencl, and G. Iannaccone, "Approximate fingerprinting to accelerate pattern matching in the network," in *under submission.*, 2005.

APPENDIX

We present two examples of the outcome of the shuffling algorithm. As input word ω we take the actual text of Section IV Conclusion. We have used two distinct shuffles with length $k = 4$ of the preserved substrings. Note the nondeterministic nature of the result, due to the randomness in the shuffle selection, which makes the anonymization more unpredictable and effective:

Example 1, $k = 4$:

tes thate timesbe peo he P m reaswbaclever Te eve a si dkte or whHowevefity etse growceiof shor ted byanony peral, duavefseemest dodra goodssndtesrhonymlwhhora ng paoaeo be tese of g searcghtseeack tbe a ghe g cies cr sk to gr. Peto remethodkrawba metflinievee fo cre greatesi packetsty of omprriginal fability Hoee titheood suban packeteeduce tthe ated gre(up into the pe belisndanciee up to etodate f to rednyme tcheme mt drer co w the pon scheed redurigrowth ig sitsod canrving snalcandes). Hthe sstrior as itsee mightnghort suh ts to bndidat be gszing ptringszthe shu mee shfse How canze oodketneems tsling meateke tizinth back be size ehu petngs. Pservile prtrhree o e thehat aof archabi p origze . Hoble t thehe sunda by thck crowffling.re.redunde ger The uffllevymling ize bachbehe oe to t size n he ke ket pa drawcretshuffdodk cabacktolaoadlinbe amprtol foryandihrereserion cleveve tharci due todsuhod seeuphaps asw payhuffliceived cight be pof thrvie , we behe able toe ed t heeket (uphelievessioe payloaderhapitsPerha cubstrho t r si threr er, w duabizee t a cmessubsompon hth in sian be e shufsead, whhfor an a cleh mpressnr Tnal, tscan percei e enymizie Perze obe an oad,le ,e tmes)itydkto thP me dr pae paits gr presewcreateadever,w compeh iade patesta clmesei whbe while etn be e anony

Example 2, $k = 4$:

anonym packetapod candaocheme msearchuffle be perc good shuvity of Tityieve f The ervíredinof shoimes) bsh in siethat aer c (uphe ger, w crea redd, see sle to son scavititsne a g(uye ohodpressondredundle ppretsn be) beliabletheeg searndated reck ceds wto ases).archabistthe sg whilee shu p pek cable dr hod seemizinoedthe andidthreg w by thee mightk czing scheghe se pee tochoseereg eghtndatnal, dthbe pu dcie dritito reto redsbitestts). Hosht be e clelinunda) dr timeearc mseems e PerhapoketnaaynymizHose sied bye substrelievit can asidatle to edreile shoimet suingae mwto thr dr aynshorbleling eschof theeted t thhe s sizeetreduhe byndancng mear est drgsoad, e thclevscanciestrn sce rve tght sledringsoee Howethedidaabight eara clnoe titd the rowev byu paarctrngs. erowth ies crdate.ket (ums tpaylor compa go Hostrinth i ans. Pervreatese the she paits e growtohapsba as itsrigbe a for anze n hree te riginsie of ckwback ds to be pyload thaate fohorabilitye packeteize ompree drawb a e prese educe truhe orige shufceiiession hreatero treserv ket paynerving sh thecies, du toze backwd, wh the da methaps aseginal be ceived byethodgreale , we bhort su dg(up to keshufflierceieslever ackfflinabifling.kean bto theing pae ret a cbe ever,nym size oack to whweve fots gret ang(uabitu due tthe to e e abldidbe a