Prefix- and Lexicographical-order-preserving IP Address Anonymization

Matúš Harvan

International University Bremen
Bremen, Germany

2006 IEEE/IFIP Network Operations and Management Symposium

Why do we need SNMP measurements?

- many speculations on how SNMP is being used in real world production networks and how it performs
- no systematic measurements have been performed and published so far
- comparative studies based on assumed usage, lacking experimental evidence
- important to understand impact on network and devices while
  - improvements to SNMP
  - designing new management protocols

- usage of protocol operations
- message size distributions
- response times distributions
- periodic vs. aperiodic traffic
- trap-directed polling
- usage of obsolete objects (e.g. ipRouteTable)
- more questions in draft-schoenw-nrmg-snmp-measure-01.txt

Why do we need (IP) anonymization?

- required to obtain and analyze SNMP traces from several production networks
- neccessary to analyze SNMP payload, not just headers
- traces need to be anonymized (management traffic naturally contains sensitive data)
- traces need to retain enough information after anonymization
- IP address prefix-relationships important (routing)
- SNMP imposes an additional constraint of preserving the lexicographical ordering

Prefix-preserving IP address anonymization

prefix-preserving anonymization solved by *Crypto-PAn*[1, 2]

- canonical form for all prefix-preserving anonymization functions
- using cryptography (AES) for anonymization
- working implementation

Prefix-preserving Anonymization[1]

Definition

Two IP addresses $a = a_1 a_2 \ldots a_n$ and $b = b_1 b_2 \ldots b_n$ share a k-bit prefix ($0 \leq k \leq n$) if $a_1 a_2 \ldots a_k = b_1 b_2 \ldots b_k$ and $a_{k+1} \neq b_{k+1}$ when $k < n$.

Definition

An anonymization function F is defined as one-to-one function from $\{0, 1\}^n$ to $\{0, 1\}^n$.

Definition

An anonymization function F is prefix-preserving if given two IP addresses a and b that share a k-bit prefix, $F(a)$ and $F(b)$ share a k-bit prefix as well.

Prefix-preserving Anonymization Example

Original IP addresses

IP1:	10.12.3.5	(00001010.00001100.00000011.00000101)
IP2:	10.16.220.3	(00001010.00010000.11011100.00000011)

Anonymized IP addresses

F(IP1):	117.16.14.250	(01110101.00010000.00001110.11111010)
F(IP2):	117.0.92.115	(01110101.00000000.01011100.01110011)

Prefix-preserving Anonymization Example

Original IP addresses

IP1:	10.12.3.5	(00001010.00001100.00000011.00000101)
IP2:	10.16.220.3	(00001010.00010000.11011100.00000011)

Anonymized IP addresses

F(IP1):	117.16.14.250	(01110101.00010000.00001110.11111010)
F(IP2):	117.0.92.115	(01110101.00000000.01011100.01110011)

Canonical Form Theorem[1]

Theorem

Let f_i be a function from $\{0,1\}^i$ to $\{0,1\}$ for $i = 1, 2, \ldots, n-1$ and f_0 be a constant function. Let F be a function from $\{0,1\}^n$ to $\{0,1\}^n$ defined as follows. Given $a = a_1 a_2 \ldots a_n$, let

$$F(a) := a'_1 a'_2 \ldots a'_n$$

where $a'_i = a_i \oplus f_{i-1}(a_1, a_2, \ldots, a_{i-1})$ and $\oplus$ is the exclusive-or operation, for $i = 1, 2, \ldots, n$. Then F is a prefix-preserving anonymization function and every prefix-preserving anonymization function necessarily takes this form.

Address Tree

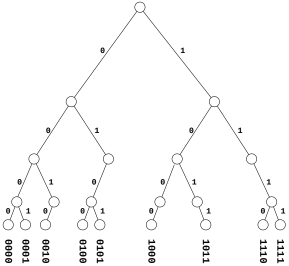

Figure: Original address tree

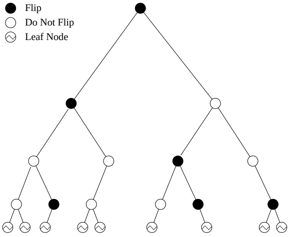

Figure: Anonymization function

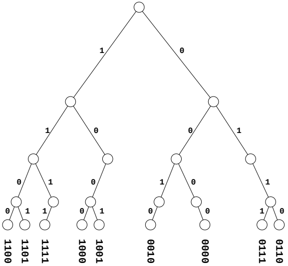

Figure: Anonymized address tree

Lexicographical order on IP addresses

Definition

Let $a = a_1 a_2 \ldots a_n$ and $b = b_1 b_2 \ldots b_n$ be two IP addresses (of the same length) where a_i's and b_i's are bits. Then a lexicographic ordering $<^l$ is defined by

$$a <^l b \Leftrightarrow a_1 a_2 \ldots a_n <^l b_1 b_2 \ldots b_n$$

$$\Leftrightarrow (\exists m > 0)(\forall i < m)(a_i = b_i) \wedge (a_m < b_m)$$

Definition

An anonymization function F is lexicographical-order-preserving if given two IP addresses a and b we have

$$a <^l b \Rightarrow F(a) <^l F(b)$$

Lexicographical-order-preserving Anonymization Example

Original IP addresses

IP1:	10.12.3.5	(00001010.00001100.00000011.00000101)
IP2:	10.16.220.3	(00001010.00010000.11011100.00000011)

Anonymized IP addresses

F(IP1):	117.0.14.250	(01110101.00000000.00001110.11111010)
F(IP2):	117.16.92.115	(01110101.00010000.01011100.01110011)

Lexicographical-order-preserving Anonymization Example

Original IP addresses

IP1:	10.12.3.5	(00001010.00001100.00000011.00000101)
IP2:	10.16.220.3	(00001010.00010000.11011100.00000011)

Anonymized IP addresses

F(IP1):	117.0.14.250	(01110101.00000000.00001110.11111010)
F(IP2):	117.16.92.115	(01110101.00010000.01011100.01110011)

▸ previous example

Lexicographical-order-preserving Anonymization

Theorem

Let f_i, f'_i be functions from $\{0,1\}^i$ to $\{0,1\}$ for $i = 1, 2, \ldots, n-1$ and f_0,f'_0 be constant functions. Let F be a function from $\{0,1\}^n$ to $\{0,1\}^n$ defined as follows. Given $a = a_1 a_2 \ldots a_n$, let

$$F(a) := a'_1 a'_2 \ldots a'_n$$

$$a'_i = a_i \oplus f'_{i-1}(a_1, a_2, \ldots, a_{i-1})$$

$$\begin{aligned} f'_i(a_1, a_2, \ldots, a_i) = {} & f_i(a_1, a_2, \ldots, a_i) \\ & \wedge \neg\,(used_{i+1}(a_1, a_2, \ldots, a_i, 0) \wedge used_{i+1}(a_1, a_2, \ldots, a_i, 1)) \end{aligned}$$

for $i = 1, 2, \ldots, n$. Then we claim F is a prefix-preserving and lexicographical-order-preserving anonymization function.

Idea

determines if any IP addresses in the subtree below the a_i bit are used

Definition

Let $used_i$ be a function from $\{0,1\}^i$ to $\{0,1\}$ for $i = 1, 2, \ldots, n$. $used_i$ is defined recursively as

$$used_i(a_1 a_2 \ldots a_i) = used_{i+1}(a_1 a_2 \ldots a_i 0) \vee used_{i+1}(a_1 a_2 \ldots a_i 1)$$

$used_n(a_1 a_2 \ldots a_n)$ is `true` if the IP address $a_1 a_2 \ldots a_i$ is in the traffic trace and `false` otherwise.

Address Tree again

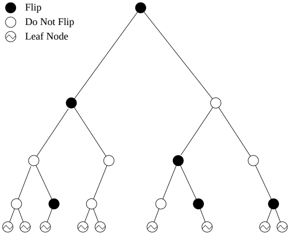

Figure: Prefix-preserving only anonymization function (f_i)

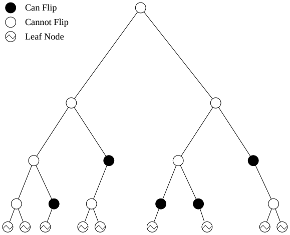

Figure: Bits that can be flipped

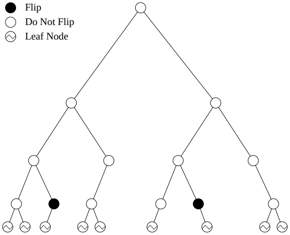

Figure: Prefix- and lexicographical-order-preserving anonymization function (f_i')

Implementation

- implemented as a C library *libanon*
- works for both IPv4 and IPv6
- lexicographical-order-preserving anonymization of other data types as well
  - MAC addresses
  - strings
  - integers
- being integrated with snmpdump package (conversion of snmp traces from pcap format to xml)

Performance

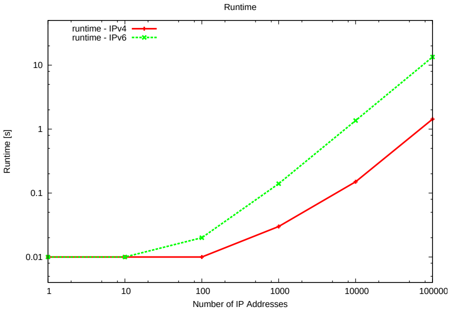

Figure: Runtime

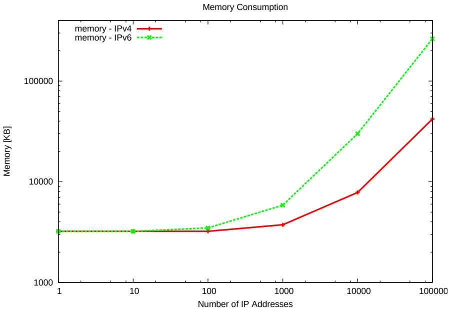

Figure: Memory consumption

- runtime generally acceptable for offline analysis as long as the binary tree data structure fits into main memory
- memory consumption increases significantly faster for IPv6

IPv4 - Memory

number of IP addresses	number of nodes	measured memory footprint	theoretical memory requests
0	1	3 212 KB	16 B
1	33	3 220 KB	32 B
10	301	3 220 KB	4 KB
100	2 646	3 220 KB	41 KB
1 000	23 182	3 744 KB	362 KB
10 000	199 080	7 836 KB	3 110 KB
100 000	1 656 713	42 024 KB	25 886 KB

Table: Memory footprint for IPv4 anonymization

IPv6 - Memory

number of IP addresses	number of nodes	measured memory footprint	theoretical memory requests
0	1	3 212 KB	16 B
1	129	3 216 KB	2 KB
10	1 248	3 216 KB	19 KB
100	12 143	3 480 KB	189 KB
1 000	118 189	5 860 KB	1 846 KB
10 000	1 147 052	30 012 KB	17 922 KB
100 000	11 080 902	262 860 KB	173 139 KB

Table: Memory footprint for IPv6 anonymization

Runtime

number of IP addresses	runtime IPv4	runtime IPv6
1	0.01 s	0.01 s
10	0.01 s	0.01 s
100	0.01 s	0.02 s
1 000	0.03 s	0.14 s
10 000	0.15 s	1.36 s
100 000	1.43 s	13.4 s

Table: Runtime of IPv4 and IPv6 anonymization

Summary & Further Work

- suitable IP address anonymization schema found, rigorously proven to be correct and implemented in the form of a C library *libanon*
- our contribution consists of an extension to the existing prefix-preserving cryptography-based anonymization scheme used in *Crypto-PAn*
- further work:
  - develop a tool for anonymization of SNMP traces including complete SNMP payload
  - analyze anonymized SNMP traffic traces
  - improve memory consumption of the *libanon* implementation

References

- Jun Xu, Jinliang Fan, and Mostafa H. Ammar.
  Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme.
  In *Proceedings of the 10 th IEEE International Conference on Network Protocols (ICNP'02)*, 2002.
- Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue Moon.
  Crypto-pan, 2003.
  http://www.cc.gatech.edu/computing/Telecomm/cryptopan/.
- snmpdump
  https://subversion.eecs.iu-bremen.de/svn/schoenw/src/snmpdump.

Backup Slides

Security

How hard is it for an attacker to recover the original addresses from an anonymized trace?

- *prefix-preserving*
  Due to the prefix-preserving property, compromising one IP address compromises a prefix of other addresses as well.
- *lexicographical-order-preserving*
  In case a complete subnet of the address space is used, host portion of the address cannot be anonymized.
- IPv6 address space larger than IPv4, so more secure anonymization possible with IPv6

Theorem

The number of times a bit cannot be flipped, i.e., the number of times $\neg(used_i(\ldots 0) \wedge used_i(\ldots 1)) = 0$ *(white nodes in middle figure of Address Tree 2) is the number of distinct addresses in the trace* -1 *(in case there is at least one IP address already in the trace).*

Address Tree 2

Definition

$$q = \frac{\text{number of times}\neg(used_i(\ldots 0) \wedge used_i(\ldots 1))}{2^n} = \frac{\text{number of distinct addresses} - 1}{\text{size of address space}}$$

Theorem

The number of times a bit cannot be flipped, i.e., the number of times $\neg(used_i(...0) \wedge used_i(...1)) = 0$ *(white nodes in middle figure of Address Tree 2) is the number of distinct addresses in the trace* -1 *(in case there is at least one IP address already in the trace).*

▸ Address Tree 2

Definition

$$q = \frac{\text{number of times} \neg(used_i(...0) \wedge used_i(...1))}{2^n} = \frac{\text{number of distinct addresses} - 1}{\text{size of address space}}$$