NAME

sc_hoiho — Holistic Orthography of Internet Hostname Observations

SYNOPSIS

sc_hoiho [**-6**] [**-d** *dump*] [**-D** *domain*] [**-f** *rtt-fudge*] [**-g** *dictionary*] [**-l** *light-speed*] [**-O** *option*] [**-r** *regex*] [**-R** *rtt-file*] [**-s** *stopid*] [**-S** *siblings*] [**-t** *threadc*] public-suffix-list.dat routers.txt

DESCRIPTION

The **sc_hoiho** utility automatically learns regular expressions that extract router names, geolocation hints, autonomous system numbers (ASNs), and autonomous system names (AS names) from hostnames. A regular expression that extracts a router name from a router hostname can be used to infer which interfaces belong to the same router (are aliases). A regular expression that extracts an ASN or AS name from a router hostname can be used to infer which AS operates the router. A regular expression that extracts geolocation hints from a router hostname can be used to infer the approximate location of the router.

To learn these regular expressions, **sc_hoiho** uses a set of training routers whose interfaces were inferred to be aliases, optionally annotated with their operating ASN. The core of the technique is described in the paper "Learning to Extract Router Names from Hostnames" published in the ACM Internet Measurement Conference (IMC) 2019. The technique that learns to extract ASNs is described in the paper "Learning to Extract and Use ASNs in Hostnames" published in IMC 2020. The technique that learns to extract geolocation hints is described in the paper "Learning to Extract Geographic Hints from Internet Router Hostnames" published in the ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT) 2021. The technique that learns to extract AS names is described in the paper "Learning Regexes to Extract Network Names from Hostnames" published in the Asian Internet Engineering Conference (AINTEC) 2021.

The supported options to **sc_hoiho** are as follows:

-6 specifies that the input training set contains IPv6 addresses, and not IPv4 addresses.

-d *dump*
specifies the dump ID to use to analyze the collected data. Currently, ID values 1 (working-set), 2 (routers), 3 (best-regex), 4 (interfaces) are valid, which (1) dump the working set of regexes for each suffix, (2) the router-level result of applying the best regexes to the training data, (3) the best regex per domain, and (4) the interface-level result of applying the best regexes to the training data, respectively.

-D *domain*
specifies the domain suffix that **sc_hoiho** should operate on. If no suffix is specified, then **sc_hoiho** operates on all suffixes.

-f *rtt-fudge*
specifies a fudge value, in milliseconds, to add to measured RTTs before evaluating if the RTT between two lat/longs is feasible. By default, this is 1ms.

-g *dictionary*
specifies a filename containing a geolocation or AS name dictionary. The format of this file is specified below.

-l *light-speed*
specifies the distance covered in metres per millisecond, and is used to evaluate if the RTT between two lat/longs is feasible. By default, this is 204.190477 metres per millisecond, the speed of light in fiber optic cable.

-O *option*

allows the behavior of **sc_hoiho** to be further tailored. The current choices for this option are:

- **application:** Show the outcome of applying the regular expressions to the application set.
- **debug:** output significant volumes of debugging information. Debugging information is only printed when a **sc_hoiho** is using a single thread.
- **json:** Output inferences using json format. This option is applicable on dump IDs 2 and 3.
- **learnalias:** infer regexes to extract router names from hostnames.
- **learnasn:** infer regexes to extract AS numbers from hostnames.
- **learnasnames:** infer regexes to extract AS names from hostnames.
- **learngeo:** infer regexes to extract geolocation hints from hostnames.
- **loadonly:** load the training data into memory and then exit.
- **noclli:** do not emit geolocation information for CLLI code prefixes.
- **noed1:** Do not infer true positives between extracted and inferred ASNs where the ASNs have an edit distance of one.
- **noip:** Do not infer the location of an embedded IP address portion, if any, in the hostname.
- **nojit:** Do not use pcre(3) or pcre2(3) just in time complication to improve regex performance.
- **norefine:** do not do any refinement of regular expressions.
- **norefine-tp, refine-tp:** do not, or do, execute the phase that builds literals into components that extract a string from the hostnames. This phase is described in phase 5.2 of the IMC 2019 paper.
- **norefine-fne, refine-fne:** do not, or do, execute the phase that embeds literals in a regex component that separates hostnames from their training routers. This phase is described in section 5.3 of the IMC 2019 paper.
- **norefine-class, refine-class:** do not, or do, execute the phase that embeds character classes in regular expressions. This phase is described in section 5.4 of the IMC 2019 paper.
- **norefine-fnu, refine-fnu:** do not, or do, execute the phase that builds additional regexes to use as a companion regex with another regex that did not cluster all interfaces on a router. This phase is described in section 5.5 of the IMC 2019 paper.
- **norefine-sets, refine-sets:** do not, or do, execute the phase that builds naming convention sets using the regexes in the working set. This phase is described in section 5.6 of the IMC 2019 paper.
- **norefine-ip, refine-ip:** do not, or do, execute the phase that builds regexes to filter hostnames with IP addresses in them, where the IP address would be part of an extracted name. This phase is described in section 5.7 of the IMC 2019 paper.
- **norefine-fp, refine-fp:** do not, or do, execute the phase that builds regexes to filter hostnames that, if matched, would result in false positives according to the training data. This phase is described in phase 5.7 of the IMC 2019 paper.
- **norefine-merge, refine-merge:** do not, or do, execute the phase that merges regular expressions that differ by a single simple string. This phase is described in section 3.3 of the IMC 2020 paper.
- **norefine-dict, refine-dict:** do not, or do, execute the phase that refines the geolocation dictionary. This phase is described in section 5.4 of the CoNEXT 2021 paper.
- **nothin:** do not remove any redundant regexes at the end of each phase, as described in section 4.5 of the IMC 2019 paper.
- **nothin-matchc, thin-matchc:** do not, or do, remove any regexes that do not meet the minimum number of matches to be considered capturing a convention. The conditions are described in section 4.5 of the IMC 2019 paper.
- **nothin-same, thin-same:** do not, or do, remove redundant regexes that make the same inferences.

- **nothin-mask, thin-mask:** do not, or do, remove redundant regexes whose inferences are entirely contained in another regex, with no additional false positives.
- **randindex:** compute the Rand Index metric on the clustering of hostnames by router name regexes according to the training data.
- **show-class:** only show hostnames where **sc_hoiho** made a classification using a regex.
- **show-good:** show regexes that **sc_hoiho** classifies as good.
- **show-promising:** show regexes that **sc_hoiho** classifies as promising.
- **show-poor:** show regexes that **sc_hoiho** classifies as poor.
- **split-locode:** identify LOCODEs that appear to be split into lengths of 2 (country-code) and 3 (location code) in hostnames.

-r *regex*
specifies the name of a file containing a working set of regexes, or a naming convention, to apply.

-R *rtt-file*
specifies the name of a file containing round trip time (RTT) measurements from systems with known locations towards routers.

-s *stop-id*
specifies the stage number to halt processing.

-S *siblings*
specifies the name of a file containing sibling ASes. Each line in the file contains a list of sibling ASes that belong to the same organization.

-t *threadc*
specifies the number of threads to use in the threadpool. By default, **sc_hoiho** will determine the number of processors online, and use all of them.

EXAMPLES

Given a set of routers in a file named routers.txt, and a copy of public_suffix_list.dat obtained from the Mozilla Foundation's https://publicsuffix.org/list/ website:

```
# node2id: 1
# node2as: 64496
192.0.2.1    esr1-ge-5-0-0.jfk2.example.net
192.0.2.10   esr1-ge-5-0-6.jfk2.example.net
192.0.31.60

# node2id: 2
# node2as: 64496
192.0.2.2    esr2-xe-4-0-0.lax.example.net
192.0.2.5    esr2-xe-4-0-1.lax.example.net
192.0.31.8

# node2id: 3
# node2as: 64496
192.0.2.6    das1-v3005.akl.example.net
192.0.2.9    das1-v3006.akl.example.net
192.0.2.44   44.2.0.192.example.net

# node2id: 4
# node2as: 64496
192.0.2.13   esr1-xe-4-0-0.lax.example.net
```

```
# node2id: 5
# node2as: 64496
192.0.2.17   esr1-xe-4-0-1.lax.example.net

# node2id: 6
# node2as: 64496
192.0.2.21   esr1-xe-4-0-1.lax.example.net

# node2id: 7
# node2as: 64500
192.0.2.25   as64500.cust.example.net

# node2id: 8
# node2as: 64501
192.0.2.29   as64501.cust.example.net

# node2id: 9
# node2as: 64502
192.0.2.33   as64502.cust.example.net

# node2id: 10
# node2as: 64503
192.0.2.37   as64503.cust.example.net
```

Then the following command will build a base set of regular expressions that extract router names, as described in section 5.1 of the IMC 2019 paper, and output the working set of regexes inferred for each suffix at the end of that phase.

sc_hoiho -O learnalias -d working-set -O norefine public_suffix_list.dat routers.txt

To obtain the best selected regular expression that extracts router names for example.net, use:

sc_hoiho -O learnalias -d best-regex -D example.net public_suffix_list.dat routers.txt

To examine how the best regular expression that extracts router names applies to the training data for example.net, use:

sc_hoiho -O learnalias -d routers -D example.net public_suffix_list.dat routers.txt

To examine how the best regular expression that extracts router names applies to the training data, as well as interfaces in the application set, use:

sc_hoiho -O learnalias -d routers -D example.net -O application public_suffix_list.dat routers.txt

To see the working set of regular expressions that extract router names built after embedding literals in captures for example.net, use:

sc_hoiho -O learnalias -d working-set -D example.net -s 2 public_suffix_list.dat routers.txt

To see how a manually-derived regular expression clusters hostnames according to the extracted router name, use:

sc_hoiho -O learnalias -d routers -D example.net -r "^([a-z]+\d+)-.+\.([a-z\d]+)\.example\.net$"
public_suffix_list.dat routers.txt

To infer regular expressions that extract ASNs from hostnames, use:

sc_hoiho -O learnasn -d best-regex public_suffix_list.dat routers.txt

To infer regular expressions that extract geohints from hostnames, and formatting the output as JSON, use:

sc_hoiho -O learngeo -d best-regex -O json -R rtts.txt -g geohints.txt public_suffix_list.dat routers.txt

HINTS

sc_hoiho can take a long time to run when inferring regular expressions that extract router names, depending on the training set involved. One option to breaking up the runtime (but not reducing it) is to capture the output from one phase, and then use that as input to the next phase. For example, to run the first three phases:

sc_hoiho -O learnalias -d working-set -s 1 public_suffix_list.dat routers.txt >phase-1.re
sc_hoiho -O learnalias -d working-set -s 2 -r phase-1.re public_suffix_list.dat routers.txt >phase-2.re
sc_hoiho -O learnalias -d working-set -s 3 -r phase-2.re public_suffix_list.dat routers.txt >phase-3.re

NOTES

sc_hoiho follows the format of the hostnames files stored in CAIDA's Internet Topology Data Kit (ITDK) which stores hostnames in lower-case, and stores characters that do not form part of the DNS's alphabet (A-Z, a-z, - and .) as a hexadecimal escaped string. For example, if a hostname contains an underscore character, such as foo_bar, then encode the underscore using the hexadecimal dictionary in `ascii`(7) as follows: foo\x5fbar.

When learning ASN regexes, **sc_hoiho** can take an optional parameter that specifies which ASNs belong to the same organization (are siblings). Each line specifies ASNs that belong to the same organization. For example, a file with the following contents:

```
64504 64505 64506
64507 64508
```

defines the ASNs operated by two organizations: one organization with ASes 64504, 64505, and 64506, and the other organization with ASes 64507 and 64508.

When learning geohint regexes, **sc_hoiho** requires parameters that specify a geohint dictionary, and RTT measurements. For the geohint dictionary, the format of the file is as follows:

```
iata code lat lng cc-st "city"
icao code lat lng cc-st "city"
clli code lat lng cc-st "city"
place "city" cc-st lat lng population
locode code lat lng cc-st "city"
facility "city" cc-st lat lng "street" "name"
country iso3166-2 iso3166-3 "name"
state cc-st "name"
vp code lat lng
```

For example:

```
iata IAD 38.9445 -77.455803 US-VA "Washington, DC"
icao KIAD 38.9445 -77.455803 US-VA "Washington, DC"
clli washdc 38.89511 -77.03637 US-DC "Washington, D.C."
place "Washington" US-DC 38.89511 -77.03637 601723
locode USIAD 38.94877 -77.4491 US-VA "Dulles Int Apt/Washington"
facility "Washington" US-DC 38.902918 -77.029149 "1275 K Street, NW" "CoreSite - Washington,DC (DC1)"
country us usa "United States"
state US-DC "Washington, D.C."
vp hlz2-nz 37.86 175.33
```

The VP geohints in the file are used by **sc_hoiho** to record where a given VP is located, and are used to identify the location associated with an RTT sample. VP geohints are not used when searching a hostname for location identifier. If the RTT file contains VP codes without a VP geohint, the VP code must match one of the other codes with a lat/lng.

For the RTT measurements, the format of each line of the file is as follows:

nodeid iata ms

For example:

N1 jfk 1
N1 iad 10
N1 yhu 20
N2 sjc 10
N2 san 12
N2 tij 16
N3 akl 2
N3 wlg 11
N3 syd 25

When learning AS name regexes, **sc_hoiho** can optionally be provided an AS name dictionary. If an AS name dictionary is not provided, then **sc_hoiho** will learn an AS name dictionary automatically. For an AS name dictionary that maps 64504 to foo, 64505 to bar, and 64506 to baz, the format of the file is as follows:

64504 foo
64505 bar
64506 baz

SEE ALSO

`pcre`(3), `pcre2`(3), `sc_ally`(1), `sc_pinger`(1), `sc_radargun`(1), `sc_speedtrap`(1),

M. Luckie, B. Huffaker, and k claffy, *Learning to Extract Router Names from Hostnames*, Proc. ACM Internet Measurement Conference (IMC) 2019.

M. Luckie, A. Marder, M. Fletcher, B. Huffaker, and k claffy, *Learning to Extract and Use ASNs in Hostnames*, Proc. ACM Internet Measurement Conference (IMC) 2020.

M. Luckie, B. Huffaker, A. Marder, Z. Bischof, M. Fletcher, and k claffy, *Learning to Extract Geographic Information from Internet Router Hostnames*, Proc. ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT) 2021.

M. Luckie, A. Marder, B. Huffaker, and k claffy, *Learning Regexes to Extract Network Names from Hostnames*, Proc. Asian Internet Engineering Conference (AINTEC) 2021.

Mozilla Foundation, *Public Suffix List*, https://publicsuffix.org/list/.

Center for Applied Internet Data Analysis (CAIDA), *Macroscopic Internet Topology Data Kit (ITDK)*, https://www.caida.org/data/internet-topology-data-kit/.

R. Govindan and H. Tangmunarunkit, *Heuristics for Internet Map Discovery*, Proc. IEEE INFOCOM 2000.

N. Spring, R. Mahajan, and D. Wetherall, *Measuring ISP topologies with Rocketfuel*, Proc. ACM SIGCOMM 2002.

A. Bender, R. Sherwood, and N. Spring, *Fixing Ally's growing pains with velocity modeling*, Proc. ACM/SIGCOMM Internet Measurement Conference 2008.

K. Keys, Y. Hyun, M. Luckie, and k claffy, *Internet-Scale IPv4 Alias Resolution with MIDAR*, IEEE/ACM Transactions on Networking 2013.

M. Luckie, R. Beverly, W. Brinkmeyer, and k claffy, *Speedtrap: Internet-scale IPv6 Alias Resolution*, Proc. ACM/SIGCOMM Internet Measurement Conference 2013.

A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, J. Smith, and k claffy, *Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale*, Proc. ACM Internet Measurement Conference 2018.

AUTHORS

sc_hoiho was written by Matthew Luckie. Marianne Fletcher added support for inferring regexes that extract ASNs and geohints from hostnames.