Passive Network Analysis Using Libtrace

Shane Alcock

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

25 November 2008

WAND Network Research Group

Outline

- Introduction and Basics
- The Libtrace Tools
- Simple Libtrace Programming
- Advanced Topics

Part Two

- Libtrace tools
    - tracepktdump
    - traceconvert
    - tracefilter
    - tracesplit
    - tracemerge
    - tracereport
    - tracestats
    - tracesummary
    - tracertstats
    - traceanon

Libtrace Tools

- Perform common passive analysis tasks
    - Splitting, merging traces
    - Dumping packets to a terminal
    - Simple statistical analysis

- Anyone can use them
    - No programming knowledge required
    - Do need to know about URIs and BPF filters

Tool Tricks

- Output trace format can differ from the input format
- Traffic filtering using BPF filter strings
    - Most tools accept a ' -f ' argument to specify a filter

Tool Tricks

- Use '–' to specify stdin / stdout in a libtrace URI
  - Context sensitive
  - e.g. `pcapfile:-`
    - Read pcap traces from stdin
    - Write a pcap trace to stdout

- Decompress traces on a separate CPU
  - `zcat trace1.erf.gz trace2.erf.gz | bfr | tracepktdump erf:-`

Tracepktdump

- Prints the contents of each packet to the terminal
  - Libtrace equivalent of tcpdump
- Decodes headers up to and including transport layer

Tracepktdump

- Usage

```
tracepktdump [-f filter] [-c count] inputURI
```

- Options

`-f filter` Output only packets that match the BPF expression

`-c count` Stop after displaying `count` packets

Tracepktdump

- Example

  Display all packets on TCP port 80

```
tracepktdump -f “tcp port 80” pcapfile:example.pcap.gz
```

Traceconvert

- Converts a trace from one capture format to another
  - Many passive analysis tools expect pcap traces
- Can only convert to formats that libtrace has write support for
  - PCAP
  - ERF
  - Linux Native
- Replay traces using PCAP or Linux Native interfaces!

Traceconvert

- Usage

```
traceconvert inputURI outputURI
```

- Example

Converting an ERF trace to PCAP

```
traceconvert erf:sample.erf.gz pcapfile:sample.pcap.gz
```

Simplistic capture from a PCAP interface to a PCAP trace file

```
traceconvert pcapint:eth1 pcapfile:capture.pcap.gz
```

Tracefilter

- Applies a BPF filter to a trace
  - Creates a new trace containing only the filtered traffic
  - The output format can be different to the input format

Tracefilter

- Usage

```
tracefilter inputURI BPFfilter outputURI
```

- Example

Create an ERF trace with only traffic to or from 192.168.2.1

```
tracefilter pcapint:eth0 “host 192.168.2.1” erf:filtered.erf.gz
```

Tracesplit

- Divide a trace into subtraces based on a particular criteria
  - Time interval, e.g. every hour
  - Number of packets, e.g. every 10000 packets
  - Size of the output file, e.g. create a series of 1GB trace files

Tracesplit

- Usage

  `tracesplit [flags] inputURI outputURI`

- Flags

`-f filter`	Only output packets that match this BPF filter
`-c count`	Split every `count` packets
`-b bytes`	Split whenever the output trace reaches `bytes` bytes in size
`-i interval`	Split every `interval` seconds of trace time
`-s start`	Start splitting at this time (UTC seconds)
`-e end`	End splitting at this time (UTC seconds)
`-m maxfiles`	Create a maximum of `maxfiles` files
`-z level`	Sets a compression level for the output traces
`-S length`	Truncate packets to at most `length` bytes

Tracesplit

- Examples

Create a single trace containing 1000 SMTP packets

```
tracesplit -f “tcp port 25” -c 1000 -m 1 pcapfile:input.pcap.gz
  pcapfile:1000smtp.pcap.gz
```

Split a single long trace into 1 hour traces

```
tracesplit -i 3600 pcapfile:input.pcap.gz pcapfile:hour
```

Grab a particular 30 minute segment from a trace

```
tracesplit -s 1228125600 -e 1228127400 pcapfile:today.pcap.gz
  pcapfile:interesting.pcap.gz
```

Tracemerge

- Merge together multiple traces into a single trace file
  - Merged packets are in chronological order
  - Input traces do NOT have to share the same capture format

Tracemerge

- Usage

```
tracemerge flags outputURI inputURI [inputURI...]
```

- Flags

`-i ifaces`	Allocate `ifaces` interfaces per input trace (ERF output only)
`-u`	Discard duplicate packets

- Example

Merging traces captured from two separate monitors

```
tracemerge -i 1 erf:merged.erf.gz erf:incoming.erf.gz
  erf:outgoing.erf.gz
```

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO

Tracereport

- Produces a set of statistical reports on a trace
  - 14 different reports
    - e.g. Port, transport protocol, TCP options
  - Can produce individual reports or the entire set
  - Each report is written to a separate file

Tracereport

- Usage

```
tracereport [flags] inputURI [inputURI...]
```

- Flags

`-f filter`	Apply a BPF filter to the input trace(s)
`-C`	Produce a report on TCP ECN
`-d`	Produce a report on packet direction
`-D`	Produce a report on dropped packets
`-e`	Produce a report on packet errors, e.g. checksum failures
`-F`	Produce a report on the number of flows
`-m`	Produce the miscellaneous data report, e.g. start/end times, PPS
`-n`	Produce a report on the network layer protocols, e.g. IP, IPv6
`-O`	Produce a report on TCP options

Tracereport

- More flags
  - `-o` Produce a report on TCP options for SYN packets only
  - `-P` Produce a report on transport layer protocols
  - `-p` Produce a report on TCP and UDP port numbers
  - `-s` Produce a report on TCP segment size
  - `-T` Produce a report on IP TOS
  - `-t` Produce a report on packet TTL

Tracereport

- Examples

Produce all the reports

```
tracereport pcapfile:input.pcap.gz
```

Produce just the reports relating to protocol / port usage

```
tracereport -P -p -n pcapfile:input.pcap.gz
```

Tracestats

- Performs simple filter-based analysis on a trace
  - Specify a filter, receive packet and byte counts at the end of each trace
- Usage

```
tracestats [flags] inputURI [inputURI ...]
```

- Flags

`-f filter`	Add a BPF filter (can be used multiple times)

Tracestats

- Examples

HTTP, SMTP and FTP traffic statistics

```
tracestats -f “tcp port 80” -f “tcp port 25” -f “tcp port 20 or
  tcp port 21” erf:sample.erf.gz pcapfile:sample2.pcap.gz
```

Simple packet size distribution

```
tracestats -f “less 500” -f “less 1000 and greater 500” -f
  “less 1500 and greater 1000” -f “greater 1500” pcapint:eth0
```

Tracesummary

- Generates a summary of the trace contents
  - Runs tracestats using filters for popular / interesting protocols

- Usage

```
tracesummary inputURI [inputURI ...]
```

- Example

```
tracesummary pcapfile:sample.pcap.gz
```

Tracertstats

- Instead of printing stats at the end, do it periodically
    - Very useful with live capture sources
    - Can be used to create time series data

Tracertstats

- Usage

```
tracertstats [flags] inputURI [inputURI ...]
```

- Flags

`-f filter` Add a BPF filter (can be used multiple times)

`-i interval` Output results every `interval` seconds

`-c count` Output results every `count` packets

`-o format` Specify a reporting output format (possible values are txt, csv and html)

Tracertstats

- Examples

Produce counts of HTTP, SMTP and UDP traffic every minute

```
tracertstats -i 60 -f “tcp port 80” -f “tcp port 25” -f “udp”
 pcapint:eth0
```

Produce counts of traffic from a particular host every 5 minutes

```
tracertstats -i 300 -f “host 192.168.2.1” pcapint:eth0
```

Traceanon

- Anonymises a trace file
  - Replaces IP addresses inside the IP header
  - Replaces checksums to be correct using the new addresses

Traceanon

- Prefix substitution
  - Replace the upper-most bits of an address with a fixed prefix
  - Simple and fast
  - Potential for address collisions in the output trace
    - e.g. substitution using 192.168.0.0/16
      - 203.106.5.101 becomes 192.168.5.101 ...
      - but 108.76.5.101 also becomes 192.168.5.101

Traceanon

- AES Cryptopan
  - No chance of collisions – address mapping is one-to-one
  - Prefix preserving – original addresses that share a prefix will still share a prefix in the anonymised trace
  - Mapping consistency is possible via the use of the same key

- More details on Cryptopan can be found at

  http://www.cc.gatech.edu/computing/Networking/projects/cryptopan/

Traceanon

- Usage

```
traceanon flags inputURI outputURI
```

- Flags

`-s`	Encrypt the source address
`-d`	Encrypt the destination address
`-c key`	Use cryptopan encryption using `key` as a key
`-p C.I.D.R/bits`	Substitute using the specified prefix

Traceanon

- Examples

Anonymise a trace using prefix substitution

```
traceanon -s -d -p 192.168.0.0/16 erf:sample.erf.gz
  erf:anon.erf.gz
```

Anonymise a trace using Cryptopan

```
traceanon -s -d -c “samplecryptopankey” erf:sample.erf.gz
  erf:anon.erf.gz
```

WAND Network Research Group
Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

www.crc.net.nz
www.wand.net.nz
www.waikato.ac.nz

WAND
Network Research Group

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

KO TE TANGATA