Tranalyzer2
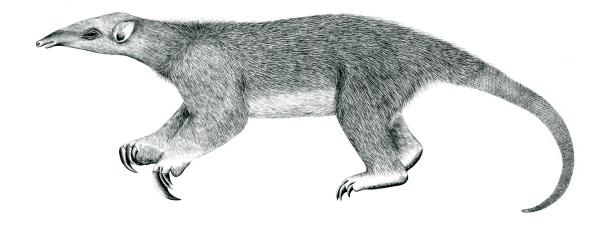
findexer

Create a binary index to quickly extract flows from pcaps

Tranalyzer Development Team

Contents

1 findexer

1.1 Description

This plugin produces a binary index mapping each flow index to its packets positions in the input pcaps. The goal of this plugin is to be able to quickly extract flows from a big pcap without having to re-process it completely. The `fextractor` tool can be used to extract flows from the pcaps using the generated index.

1.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
FNDXR_SPLIT	1	Split the findexer file with t2 `-W` option
FNDXR_SUFFIX	`"_flows.xer"`	Suffix for flows output file
FNDXR_PKTSXER_SUFFIX	`"_packets.xer"`	Suffix for packets output file

1.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- FNDXR_SUFFIX

- FNDXR_PKTSXER_SUFFIX

1.3 fextractor

The `fextractor` tool can be used to extract flows using the generated `_flows.xer` index.

```
Usage: fextractor -r INPUT[:start][,end] (-w OUTPUT | -n) [OPTIONS]... \
          [[DIR@]FLOWINDEX[:start][,end]]... | [PKTNO]...

Extract the flows FLOWINDEX using the _flows.xer INTPUT generated by Tranalyzer2 findexer plugin.
Alternatively use a list of findexer files generated by Tranalyzer2 -W option from index start
to end. The extracted flows are written to the OUTPUT pcap.

An optional packet range can be provided on each command line FLOWINDEX to only extract packets
in the range [start, end] of this flow. If start or end are omitted, they are replaced by,
respectively, the first and the last available packets in the flow. The FLOWINDEX can also
optionally be prefixed with a direction A or B, by default both directions are extracted.

When using packet mode (-P), a optional list of packet numbers (PKTNO) can be provided instead of
the flow indexes (FLOWINDEX).

OPTIONS:
  -P        packet mode, extract specific packets (from "t2 -s" output) instead of whole flows.
  -r INPUT[:start][,end]
            either read packet indexes from a single _flows.xer file named INPUT
            or read packet indexes from multiple _flows.xer files prefixed by INPUT
            and with suffix in range [start, end]. If start or end are omitted,
```

```
                  they are replaced by, respectively, first and last available XER files.
                  When using packet mode (-P), INPUT must instead be in _packets.xer format.
     -w OUTPUT  write packets to pcap file OUTPUT
                  OUTPUT "-" means that the PCAP is written to stdout.
     -f         overwrite OUTPUT if it already exists
     -i FILE    read flow indexes / packet numbers from FILE. FILE can either be in _flows.txt
                  format (flow index in 2nd tab-separated column), have one flow index per line,
                  or be in _packets.txt format (packet number in 1st tab-separated column) when
                  using packet mode (-P).
                  FILE "-" means that flows are read from stdin.
     -b         by default when FILE is in _flows.txt format, only directions present in
                  it are extracted, this option forces both directions to be extracted even if
                  only the A or B direction is present in the flow file.
     -s N       skip the first N PCAPs
     -p DIR     search pcaps in DIR
                  should only be set if pcaps were moved since Tranalyzer2 was run
     -n         print oldest PCAP still available, its first packet timestamp and exit
     -h         print this help message and exit
```

Example to extract flow 42, 123 and 1337 to the `output.pcap` file:

```
    fextractor -r ~/t2_output/dmp1_flows.xer -w output.pcap 42 123 1337
```

Example to extract packet 100, 150 and 200 to the `output.pcap` file:

```
    fextractor -P -r ~/t2_output/dmp1_packets.xer -w output.pcap 100 150 200
```

1.4 Example scenario

We want to extract all the flows whose source or destination are in China, to look at them in Wireshark.

First, we run tranalyzer with at least the findexer , `basicFlow` and `txtSink` plugins. The findexer plugin will generate a `_flows.xer` index file which keeps a list of packets positions in the original PCAP for each flow.

```
    [user@machine]$ tranalyzer -r capture01.pcap -w t2_output/capture01
```

We now use the `srcIPCC` and `dstIPCC` columns to filter flows with IPs in China.

```
    [user@machine]$ grep IPCC t2_output/capture01_headers.txt
    9      SC:N    srcIPCC Source IP country code
    12     SC:N    dstIPCC Destination IP country code
```

The country code are in the 9 and 12 columns. The flows to extract can directly be piped to the `fextractor` which then pipe the extracted PCAP to Wireshark.

```
    [user@machine]$ awk -F"\t" '$9 == "cn" || $12 == "cn"' t2_output/capture01_flows.txt | \
            fextractor -i - -r t2_output/capture01_flows.xer -w - | wireshark -k -i -
```

By using `tawk` we don't even need to look at the column numbers in the header file, we can directly extract the flows of interest using the column names. `tawk` also provides a `-k` option which takes care of extracting the flows and opening them in Wireshark.

```
[user@machine]$ tawk -k '$srcIPCC == "cn" || $dstIPCC == "cn"' t2_output/capture01_flows.txt
```

1.4.1   Extract specific packets from packet forensic mode

We want to extract all packets containing the string "Apache" to an `apache.pcap` file.

```
[user@machine]$ t2 -s -r capture01.pcap -w t2_output/capture01
[user@machine]$ tawk '$l7Content ~ /Apache/' t2_output/capture01_packets.txt | \
        fextractor -P -i - -r t2_output/capture01_packets.xer -w apache.pcap
```

1.5   Additional Output (findexer v2)

1.5.1   Flows binary index

A binary index with suffix `_flows.xer` is generated. This file is composed of the following sections in any order, except the flows findexer header which is always at the beginning of the file. All numbers are written in little endian.

flows findexer header

```
struct findexer_header {
    uint64_t magic;           // 0x32455845444e4946 = FINDEXE2
    uint32_t pcapCount;       // number of input pcaps provided to tranalyzer
                              // 1 if -r or number of lines in -R file
    uin64_t firstPcapHeader;  // offset of the fist pcap headers (see next section)
                              // in the _flows.xer file
};
```

flows pcap header

```
struct pcap_header {
    uint64_t nextPcapHeader;  // offset of the next pcap header
                              // in the _flows.xer file
    uint64_t flowCount;       // number of flows in this pcap
    uint64_t firstFlowHeader; // offset of the first flow header (see next section)
                              // of this pcap in the _flows.xer file
    uint16_t pathLength;      // length of the path string
    char* pcapPath;           // path string (NOT null terminated)
};
```

flow header

```
struct flow_header {
    uint64_t nextFlowHeader; // offset of the next flow header
                             // in the _flows.xer file
```

```
    uint64_t flowIndex;      // Tranalyzer flow index (2nd column in flow file)
    uint8_t flags;           // flow flags (see next section)
    uint64_t packetCount;    // number of packets in this flow
#FOREACH packet in the flow
    uin64t_t offset;         // offset in the pcap where to find the packet
#ENDFOREACH
};
```

flow flags

flags	Description
2^0 (=0x01)	This is a B flow.
2^1 (=0x02)	This is the first XER file in which this flow appears.
2^2 (=0x04)	This is the last XER file in which this flow appears.
2^3 (=0x08)	and all higher values: reserved for future use.

1.5.2 Packets binary index

A binary index with suffix `_packets.xer` is generated when tranalyzer is run in packet forensics mode (`-s`). This file is composed of the following sections in any order, except the packets findexer header which is always at the beginning of the file. All numbers are written in little endian.

packets findexer header

```
struct pktsxer_header {
    uint64_t magic;          // 0x3252455853544b50 = PKTSXER2
    uint32_t pcapCount;      // number of input pcaps provided to tranalyzer
                             // 1 if -r or number of lines in -R file
    uin64_t firstPcapHeader; // offset of the fist pcap headers (see next section)
                             // in the _packets.xer file
};
```

packets pcap header

```
struct pkt_pcap_header {
    uint64_t nextPcapHeader; // offset of the next pcap header
                             // in the _packets.xer file
    uint64_t first_pkt;      // fist PKTNO in this PCAP
    uint64_t last_pkt;       // last PKTNO in this PCAP
    uint16_t pathLength;     // length of the path string
    char* pcapPath;          // path string (NOT null terminated)
#FOREACH packet in the PCAP
    uin64t_t offset;         // offset in the pcap where to find the packet
#ENDFOREACH
};
```

1.6 Limitations

- PcapNg format is not supported (packet offsets in the pcap cannot be computed because of the additional block structures). PcapNg can however be converted in standard Pcap using the following command:

```
editcap -F pcap input.pcapng output.pcap
```

- The findexer file cannot be generated when a BPF is used. With a BPF, not all packets are processed by Tranalyzer2 which makes it impossible to compute packets offsets in a PCAP.

1.7 Old format (findexer v1)

findexer header

```
struct findexer_header {
    uint64_t magic;          // 0x52455845444e4946 = FINDEXER
    uint32_t pcapCount;      // number of input pcaps provided to tranalyzer
                             // 1 if -r or number of lines in -R file
#FOREACH pcap
    uin64_t pcapHeaderOffset; // offset of the per pcap headers (see next section)
                             // in the _flows.xer file
#ENDFOREACH
};
```

pcap header

```
struct pcap_header {
    uint16_t pathLength;    // length of the path string
    char* pcapPath;         // path string (NOT null terminated)
    uint64_t flowCount;     // number of flows in this pcap
#FOREACH flow
    uint64_t flowIndex;     // Tranalyzer flow index (2nd column in flow file)
    uint64_t packetCount;   // number of packets in this flow
    uint64_t packetsOffset; // offset in the _flows.xer file where this flow packet
                            // offsets in the pcap (see next section) are located
#ENDFOREACH
};
```

packet offsets

```
#FOREACH packet in the flow
    uin64_t offset; // offset in the pcap where to find the packet
#ENDFOREACH
```

To extract flow `123`, the following steps are followed:

- open the `_flows.xer` file and check it has the right `magic` value

- for each pcap in `pcapCount`

    - read the pcap header located at `pcapHeaderOffset` in the `_find.xer` file.

- for each flow in `flowCount`
  * if `flowIndex == 123`: read `packetCount` offsets at position `packetsOffset` in the `_flows.xer` file and extract packets located at these offsets in the pcap at `pcapPath`