Tranalyzer: Versatile High Performance Network Traffic Analyser

Stefan Burschka Benoît Dupasquier
Department of Traffic Mining and Network Security
RUAG Schweiz AG – RUAG Defence, Bern, Switzerland
firstName.lastName@ruag.com
sburschka01@qub.ac.uk bdupasquier01@qub.ac.uk

***Abstract*—IP-based networks are prone to hardware failures, software errors and misconfigurations. This leads to service outages, such as those experienced by American Airlines in 2015. Moreover, cyber threats are becoming ever more sophisticated. As demonstrated by recent success stories of malware, such as the crimeware BlackEnergy, current defence solutions are insufficient to detect those anomalies and threats. Indeed, the widespread use of cryptography and obfuscation techniques limits the effectiveness of standard solutions relying on content inspection. Although statistical based approaches are able to deal with some of these limitations, threats such as data exfiltration and covert channels remain challenging to detect. This paper presents Tranalyzer, a flow-based traffic analyser built upon a flexible plugin-based architecture, allowing efficient processing and analysis of network traffic. The program is presented through a series of real-life scenarios dealing with traffic mining and troubleshooting and provides the analyst with a methodology on how to tackle such challenges, even when encryption or obfuscation techniques are being used.**

***Index Terms*—Network Security; Troubleshooting; Traffic Forensics; Preprocessing; Data Mining; Big Data; Open Source.**

I. Introduction

Ubiquitous computing and technological advances have resulted in a dramatic increase in network traffic. With ever more networked devices, IP infrastructures continue to become larger and more complex, resulting in stability and security issues. Policies such as bring-your-own-device (BYOD) and the presence of legacy software and hardware (HW) increase risks. Such problems can be alleviated by monitoring and analysing network traffic. High speed networks, encryption and huge volume of data render this task even more complex. Data needs to be aggregated and reduced to a manageable level. The challenge is to preserve useful features which could potentially help in detecting anomalies, such as misconfiguration problems, attacks or malicious traffic. An established solution is flow based aggregation [1], [2] using only layer 3 and 4 features, such as the standard five-tuple, i.e., source and destination IP/port and protocol.

Nevertheless, experience in real life troubleshooting and security operations of corporate, SCADA and operator networks has revealed several shortcomings of existing tools, such as the need for more flexible aggregation, e.g., using the VLAN ID, network masks and layer 2 header features. In addition, important aspects, such as scalability, performance and feature provision for anomaly detection or traffic mining (TM) applications are not always met. To allow both researchers and network administrators to analyse high volume traffic on- and off-line, Tranalyzer 2 (T2), a lightweight flow analyser written in C and containing a set of plug-ins for troubleshooting and TM has been introduced to the open source community [3], [4]. Built atop the libpcap [5] library, T2 accepts not only IPv4/6, but also layer 2 and encapsulated packets, such as MPLS, L2TP and GRE, from standard pcap files or live interfaces. It is a memory-efficient flow aggregator which facilitates the development of plug-ins by a well defined API.

The output is available in text or binary format that can be used for further post-processing, e.g., using simple bash commands or complex third party programs. In addition, database, socket or even JSON output can be generated, rendering the integration of T2 with existing Elasticsearch and Kibana solutions seamless. Packet-based output, similar to TShark, complements flow output and is an invaluable drill-down feature.

The main focus is to provide security analysts and troubleshooters with a methodology on how to perform traffic mining, even in the presence of encryption or obfuscation. Tranalyzer has been used in the past to solve TM problems, such as SSH command guessing [6] or content guessing from encrypted VoIP [7], and gaining new functionalities after every analysis. The practical knowledge gained from these tasks has led to the definition of an efficient work flow, starting with large datasets beyond the terabyte range, and drilling down from flows to specific packets which can further be analysed using existing packet-based solutions such as Wireshark [8] or tcpdump [5].

This paper is organised as follows: Section II discusses related work and similar tools. Section III explains the concept, basic architecture and functionality of Tranalyzer. The different plug-ins, their output and application is also reviewed. Section IV discusses some real-life traffic mining problems on large complex datasets that were applied to T2. Section V compares the performance of T2 against similar products. Section VI summarises this work and highlights directions for future work.

II. Related Work

In this section, several network analysis tools proposing flow or packet-based analysis are reviewed.

Cisco NetFlow [9] is widely used and a de facto standard in monitoring. It provides flexible flow aggregation through masking of IP, ports, layer 2 interfaces and VLAN. It relies on dedicated HW, which can be expensive and cannot be easily extended to support configurable statistical features per flow and AI classifiers. Tranalyzer was inspired by NetFlow and supplies an IPFIX output plugin to serve NetFlow collector tools such as QRadar [10], nfdump [11] or SSHCure [12].

sFlow [13] is an industry standard for sampled flow based statistics. Similar to Cisco Netflow, it is built for high speed traffic statistics and troubleshooting in cooperation with existing switches. L7 payload of all packets and packet statistics per flow, necessary for encrypted TM, are not readily available. L7 extraction support is also not supported.

Bro [14] is a highly capable and flexible network security monitor providing flow based output similar to T2. It also serves the forensics area by allowing layer 7 extraction, such as HTTP and VoIP content, thus making it a good candidate for a performance comparison. However, it lacks statistical features for TM and troubleshooting, because it was not initially designed for such tasks.

Tstat [15] has many years of experience in the troubleshooting area and provides a rich set of very useful features, along with support for Bayesian classification and RRD [16] monitoring. However, it lacks support for various types of encapsulations, such as L2TP and MPLS, thus preventing a performance comparison with Tranalyzer.

SiLK [17] is a collection of open source network traffic analysis tools developed by the CERT NetSA. The output is quite limited and matches that provided by the *basicFlow* and *basicStats* plug-ins of T2, but lacking many statistical features and anomaly flags. Furthermore, the custom binary output format requires specific utilities to perform tasks as simple as cut, cat, sort or count, while T2 relies on the tried and trusted equivalent UNIX utilities which, combined with AWK, prove far more powerful and flexible.

Wireshark [8] is a powerful packet oriented traffic analyser with support for flow-based analysis. Although extremely useful for investigating specific packets, this open-source tool is not practical when dealing with "large" dumps (several GB), where memory usage and loading time become prohibitive. It has one of the most extensive list of protocol dissectors and is able to extract layer 7 content. TShark, the command line version of Wireshark, does not provide all the functionality of its GUI counterpart, e.g., content extraction of SMB2 or FTP traffic is not readily available.

Snort [18] is a well known open source Intrusion Detection System (IDS). However, it has no TM capabilities, its main task being searching for regular expressions at the packet or cross-packet level.

TIE [19] is a 5-tuple flow based traffic classification engine also implementing several AI based traffic classification algorithms. Unfortunately, it is not fully available as open source and most importantly, it does not support operator and corporate characteristic traffic encapsulation. In addition, according to the website, most of the plugins are not in a stable state. Troubleshooting and forensic support was also elusive.

Haddadi et al. [20] compare the ability of features provided by five open source tools, namely Maji, YAF, Softflowd, T2 and Netmate, to detect botnets, such as Zeus and Conficker. Results show that T2 combined with the C4.5 algorithm outperforms the others in terms of detection rate and false positive rate, even though only numeric fields were used.

To conclude, although several network traffic analysis tools exist, few of them meet all the requirements for practical TM, troubleshooting and security application in real corporate or operator environment. Some important aspects are flexibility, simplicity, support for encapsulation, speed and easy to process output. For example, most of the tools do not support encapsulations, such as L2TP or MPLS, and only use the standard five tuple for flow aggregation, which limits their usefulness in telco environments. Moreover, important forensic features, e.g., the link between flows, packets and content is mostly not available, except for Bro, TShark and Wireshark. Hence, those tools will be used for a performance comparison with Tranalyzer 2.

III. T2 Architecture

This section presents an overview of T2 architecture, functionality and usage.

A. Basic Functionality and Organization

Tranalyzer consists of a core and a set of plug-ins which can be activated depending on the user needs. The core performs flow aggregation and management, while the plugins receive notifications when a flow is created, terminated or timed-out and when a new packet is available. The global flow architecture is depicted in Fig. 1.

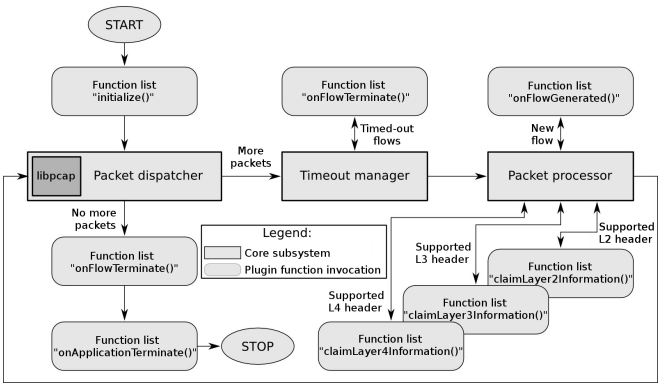

Fig. 1. Tranalyzer functional diagram

Although T2 is flow-based, it also features a packet mode, similar to TShark, but easier to parse and providing a unique numerical ID linking every packet to its flow. In addition, an alarm mode allows Tranalyzer to act as an IDS, only generating output when requested by one or more plugins,

such as AI classifiers or regular expressions matches. The force mode enables each plugin to terminate flows on demand if certain conditions are met, such as counter overflow. The monitoring mode produces configurable key parameters at regular intervals which can be fed into tools such as RRD. All modes can operate concurrently. Professional network operations demand high speed and stable execution. Therefore, most of the configuration options are stored in a header file and require recompilation if changed. Although this approach might seem cumbersome at first, the gain in speed makes this effort worthwhile. Furthermore, thanks to the concept of plug-ins, T2 can be easily and efficiently tailored and optimised to the task at hand. A nifty feature is the ability to bind Tranalyzer to a single core in a multicore processor. This can be used to run several T2 in parallel using different configurations, e.g., IPv4, IPv6 or layer 2 flows.

To summarise, T2 performs the following main tasks:

- Packet capture,
- Packet-to-flow allocation,
- Flow timeout handling,
- Plug-in function invocation,
- Flow/packet based output formats.

B. Packet-to-Flow Allocation

Standard flow-based analysis tools rely on a standard five tuple, namely source/destination IP and port, and protocol number. When working with large corporate networks, the need for a more general concept becomes apparent. Therefore, Tranalyzer proposes to extend the five tuple to a six to nine tuple, where the sixth element is the VLAN ID and the tuple above are the MAC addresses and the EtherType. For added flexibility, source and destination addresses can be masked to identify subnets instead of hosts, fields can be ignored and port ranges can be aggregated into a single flow.

The benefits of such aggregations are manifold. First, the number of flows and therefore the size of the output is reduced. Second, the aggregation provides a higher level overview of the network operations. In addition, subnetworks can be labelled according to user defined subnet numbers, enabling the user to rapidly identify relevant traffic flow clusters in high volume flow files.

Finally, contrary to many existing tools, Tranalyzer is aware of the flow direction, e.g., client $\rightarrow$ server and server $\rightarrow$ client, and labels it as `A` and `B` flows, respectively. Moreover, the initiator of the connection is detected and different flows can be linked together, e.g., FTP communication and data flows or ICMP message and flow which caused the message. This facilitates connection oriented traffic analysis.

The next section presents an overview of the different plug-ins available.

C. Plug-ins

The open source version of Tranalyzer comes with the following plug-ins and functionalities:

- Statistics about ICMP traffic and flow tracking,
- Geo-localization of IP addresses,
- MAC addresses and manufacturers,
- Classification based on nDPI [21] or port,
- Basic and descriptive statistics,
- Connection counter (Section IV-C)
- Statistics based on the N first packets,
- TCP state machine and flags analyser,
- Text, binary and JSON output.

Some of the currently closed source plugins include:

- Web, email, routing, name resolution and VoIP decoders and analysers,
- Covert channels detector,
- Password extractor,
- Regular expressions,
- Math and AI: centrality (Section IV-D), entropy, wavelet, bayes, ESOM (Section IV-G)
- Database, TCP/UDP Socket and Netflow output.

In addition, a set of scripts to convert the output to a variety of open source tools, such as RRD [16] or Gnuplot are provided.

T2 supports easy plugin development by supplying all pointers to dissected protocol layers and packet features, including flow timeout. This timeout, can be changed at each packet event, facilitating the implementation of state machines for protocol anomaly detection. Furthermore, features produced by one plugin can be shared with subsequent plugins. For a comprehensive list of plugins, scripts and features refer to [3].

The next section discusses various real-life scenarios and how Tranalyzer was used to solve those challenging cases.

IV. Scenarios

In order to illustrate the potential of Tranalyzer, some practical cases are shortly discussed. All cases have in common the traffic mining methodology which is independent of the L7 content and thus encryption. It involves a top down procedure starting with the initial end report of T2, providing global situational awareness, e.g., data acquisition problems and insights into malicious activities. The next step consists of evaluating the global statistical summaries for layer 4 protocols, detailed statistics about ICMP messages and port distribution for TCP, UDP and SCTP traffic. This provides an overview of the type of traffic and its relevance and directs the analyst towards specific questions, such as “biggest talker”, protocol or connection anomalies or QoS problems.

These questions can be formulated using bash commands or command line based languages, such as AWK or Perl. This is a recursive process, which ends when the few relevant flows are detected. Using this reduced set of flows, further drill-down to the packet level can be performed.

In practice, T2 has been successfully used in:

- Troubleshooting,
- Operational picture and monitoring,
- Traffic classification,
- Malware and anomaly detection,
- Forensics and content extraction.

A. Troubleshooting

Troubleshooting large critical infrastructures often requires analysing several TB of traffic data in order to find the source of a problem. In addition, the cause of the problem might be unknown and no information about the data acquisition might be available, e.g., due to privacy issues. Therefore, T2 labels faults happening during data acquisition, such as insufficient snap length, time synchronization problems, MTU and QoS problems. Global warnings about the quality of the data are issued and potentially corrupt flows are flagged, providing a measure of relevance and trust, not only for the whole pcap, but also for every flow. In addition, the effective bandwidth and the average number of flows per second are calculated, providing additional information about the soundness of the data.

Furthermore, certain statistical features, such as the number of unique IP talkers and listeners, provide an insight into the topology of the network under scrutiny. Distinguishing sub-networks from one another is also a key factor for rapid understanding of the traffic flow and to categorize the importance of the traffic. Hence, T2 offers the automatic labelling of flows according to user defined subnet numbers.

A lot of network and application problems can be found by passively evaluating round-trip measurements [22]. The *tcpFlags* plugin measures trip and round-trip times (TT and RTT) for all layer 4 protocols as well as the jitter ($\sqrt{J}$) acquired from the point of traffic acquisition. Moreover, several TCP related parameters are produced, such as window size evolution, SEQ/ACK number faults, maximum segment size (MSS), boot time, etc.

This enables the troubleshooter to estimate a maximum possible bandwidth per machine and even per application [23], [24]. Especially voice applications, such as Skype or SIP/RTP, can thus be assessed on a flow basis, even when RTCP is not present. Specific plugins are available to extract and compute QoS parameters.

The following section explains how T2 estimates the round-trip time and the jitter.

1) Trip and round trip time estimation: The basis of the round-trip time and the jitter is the estimation of a correct trip-time between two hosts x and y during the lifetime of the corresponding A or B flows separately. In order to remove the reaction times of the hosts and the human operator, the SYN-SYN/ACK-ACK sequence can be evaluated, if TCP traffic is present. Nevertheless for certain protocols, where human interaction is not present and program reaction times are minimal, e.g., P2P, voice, video, etc., a continuous TT estimation during the flow proves useful to pinpoint problems at the host, application or network level. Then, the TT can be calculated during flow life time using a modified infinite impulse response (IIR) first order filter. To assure the same accuracy as if calculated at the end of the flow from a complete distribution, the IIR filter constant has to become flexible on a packet per packet basis.

Let $i \in \mathbb{N}$ be the number of packets sent from host x to host y (A flow) or from y to x (B flow). Let $TT(x,y)_i$ be the estimated trip time between x and y, and Δt_{i+1} the time difference between the last packet i received in the A flow and the new one $i+1$ acquired in the B flow. Then, the new estimated trip time in the B flow is computed as:

$$TT(x,y)_{i+1} = \left(1 - \frac{1}{i}\right) TT(x,y)_i + \frac{\Delta t(x,y)_{i+1}}{i} \quad (1)$$

This leads to the following equation for the RTT:

$$RTT_{i+1} = TT(x,y)_{i+1} + TT(y,x)_{i+1} \quad (2)$$

For TCP traffic, the difference between the SYN-SYN/ACK-ACK sequence and the RTT IIR estimate in combination with the window size evolution statistics helps to find application and network problems. For voice applications, the jitter $\sqrt{J}$ is an important parameter. It can be estimated from the IIR filtered variance J derived from TT:

$$J(x,y)_{i+1} = \left(1 - \frac{1}{i}\right) J(x,y)_i + \frac{\left(\Delta t_{i+1} - TT(x,y)_{i+1}\right)^2}{i} \quad (3)$$

The corresponding RTT jitter variance is then:

$$J_{i+1} = J(x,y)_{i+1} + J(y,x)_{i+1} \quad (4)$$

In practice, the quality of the J measure was confirmed by acquiring RTCP jitter packet information from the SIP clients in an operator network.

2) The Invisible TCP Flows: A multimedia hosting centre for 5000 users reported a sudden disruption of certain office services and unmotivated delays, although the network throughput was generally excellent. T2 was applied to a prominent entry point, a firewall, supervising all ingress and egress traffic. The final T2 report of a 11TB pcap revealed that protocols such as SMB and MS SQL had a large global packet flow asymmetry of 0.76 (Eq. 5).

$$A = \frac{(pktsA - pktsB)}{(pktsA + pktsB)} \quad (5)$$

As TCP requires constant two-way communication, such an asymmetry is a very suspicious result if all traffic is present in the pcaps. By sorting the flow file according to the biggest talker and to the highest packet flow asymmetry, established TCP flows with packets and bytes asymmetry of +1.0 emerged. Hence, established unidirectional TCP flows.

The *tcpFlags* plugin sums all differences between ACK numbers, revealing that not all opposite flows were seen. The large amount suggested a routing problem. After checking the said firewall and change logs, a misconfiguration was detected, blocking an outbound high port range. The usage of OSPF dependent routing the existence of a low performing gateway bridging the firewall from another section of the network via the Internet back to the customers revealed finally the nature of the anomaly.

3) Unexplained Response Times: A deadlock on a productive Oracle DB in a large call centre produced substantial and random delays for the agents. The application of T2 to a 6TB pcap gathered over two weeks showed unusually high amount of TCP flows timed out and matched the reports of the agents. By simply sorting the flow file for the biggest talkers in term of packets and bytes transmitted and destination connection count per source IP, the Oracle server was instantly top listed. The suspicion was confirmed by the TCP anomaly flags showing multiple retries directed to the same destination (the DB) combined with several flows being timed out. Finally, the Oracle server logs revealed that both the client and server program submitted an unnecessary large amount of SQL code to the DB ending in spurious deadlocks, causing the experienced long delays and interruptions.

4) Load Balancing and Failover: In order to effectively troubleshoot load balancing or failover problems the plugin *macRecorder* tracks, for each flow, all layer 2 MAC address pairs present during the lifetime of the flow, including packet counts for each layer 2 path. A HW failure at a telecom operator could be detected by calculating an overall MAC pair packet asymmetry over multiple flows, similar to Equation 5.

B. ARP Spoofing

T2 is able to create pure layer 2 flows. Thus, a well known tool `arpwatch` [25] could be replaced by one plugin, namely *arpDecode*, dissecting all information about ARP messages per flows including anomaly bits for all kinds of ARP spoofing. In combination with the alarm mode, only flows which might indicate ARP spoofing are released, thus achieving similar functionality as `arpwatch`.

C. Connection Anomaly Detection

A prominent security problem is the horizontal spread of malware in a corporate network normally undetected by perimeter defence. The easiest give-aways are anomalies in host traffic patterns, which can be readily discovered by looking at the global statistics file. A flat and even global port distribution or the evolution of host connections per flow/time are efficient methods to assess the role of a machine. The latter produces a simple and intuitive anomaly measure, namely the number of connections from source to destination IP and between source and destination IP during the lifetime of a flow. By combining the flow creation time, source IP and source IP connection value in a 3D-waterfall plot, the evolution of a whole network can be monitored. As depicted in Fig. 2, a botnet clearly reveals its nature over time on the left side of the IP axis, while smaller peaks on the right side denote P2P streams. All the normal client traffic shows insignificant low counts.

D. Centrality

The connection anomaly detection problem is better addressed by the so called centrality, a principal component analysis (PCA) of the connection matrix containing either binary values or any other flow variable, such as the number

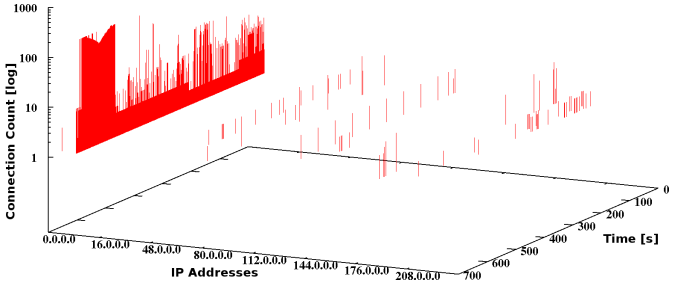

Fig. 2. connectionCounter plugin time series

of flows, packets or bytes transmitted between two peers. The type of entries in the connection matrix can be chosen depending on what anomaly is being targeted. For example, binary entries are good detectors for network anomalies and number of flows are more appropriate for malware detection. The largest eigenvector of the connection matrix represents the importance of an IP in the overall layer 3 network.

The *centrality* plugin computes this matrix every N seconds and produces an output, which can be post-processed by an AWK script generating the waterfall graph depicted in Fig. 3.

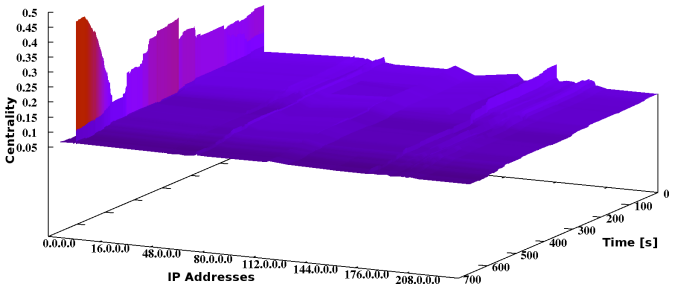

Fig. 3. centrality plugin time series, Number of flows/IP pair

Centrality reveals a steadier and more discriminable anomaly signature than the more volatile one generated by the connection plot (Fig. 2). The walls on the left with highest centrality denote IPs/clients with malware operation. The floor denotes normal clients, while the scattered smaller walls on the right denote P2P clients. A publication about centrality is in preparation.

E. Packet Signal Processing

Nowadays, communication via Internet is generally encrypted. Some applications even use layer 7 protocols as covert transport. This limits the effectiveness of standard approaches, as content cannot be inspected. Nevertheless, statistics such as packet length (PL) and inter-arrival time (IAT) can be used to classify such encrypted data. Even information regarding the payload can still be inferred. Many application programmers use existing cryptographic and audio libraries. These third party products can influence layer 4 features, such as PL and IAT.

In addition, most protocols are based on a state machine. The different states can often be distinguished in the packet stream. The initialisation phase is often short (the first 10

packets) and can be used as a signature to identify certain protocols or to infer information about the nature of the encrypted traffic [26], [27]. The following longer phase consists of the actual (encrypted) content, e.g., SSH [6] or Skype [7], which can actually be estimated by applying methods of signal processing, such as scalar Kalman filters [7] or cepstrum analysis, provided that the maximum necessary bandwidth of the signal is properly estimated and correctly sampled [28].

F. Statistical Anomaly Detection

Aggregating the PL and IAT information of a flow into a two-dimensional histogram serves as a signature for statistical TM methods. The shape of the PL distribution alone is very specific for the type and content of the traffic, thus training supervised AI classifiers such as ANN or Bayes yield good results [26], [27]. T2 supplies such a configurable distribution for each flow, which can be accessed by any subsequent plugin. All high order descriptive statistics for traffic mining depend on this distribution. A sample distribution of a HTTP tunnelled Skype distribution is depicted in Fig. 4.

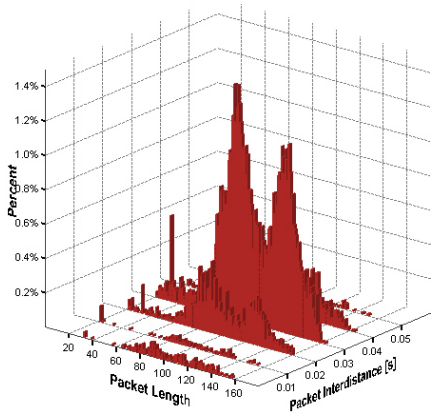

Fig. 4. 3D PL and IAT distribution of Skype

The non-uniform shape of the distribution suggests that Skype traffic can be statistically identified and information about its content is leaked [7]. To prevent such statistical fingerprinting, developers of multimedia applications should aim towards a uniform distribution of the PL-IAT signature, e.g., by using sophisticated padding. The PL-IAT distribution also serves as the basis for the *descriptiveStats* plugins, used for unsupervised TM of unknown traffic (Section IV-G).

G. Anomaly Detection using ESOM

For encrypted traffic containing unknown problems, a high performance Kohonen type emergent self-organizing map (ESOM) [29] can be trained in an unsupervised way. The resulting model can be loaded by the *esomClassifier* plugin, serving as a black box classifier. The training process creates a 2D anomaly visual chart using, e.g., the multidimensional output of the *descriptiveStats* plugin. Each point in the map represents one or several flows, clustered according to their statistical similarity (Fig. 5). A high level structural analysis can be conducted to detect unusual patterns such as "wings" (botnet), worms (P2P) or a few isolated elevated dots (unauthorized DNS zone transfer) in normal "green" traffic (browsing, email, etc.). Often the regions have to be investigated in more detail. Thus, a drill down to the flow information behind each dot or group of dots is possible. Even a forensic drill down to the packet level is available, with the possibility to automatically load selected flows/packets into Wireshark. A detailed description of all features of the ESOM is beyond the scope of this paper. For more detailed information, the reader is referred to [30].

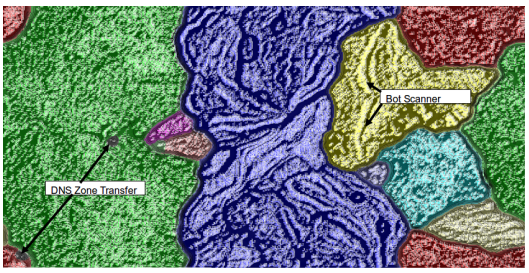

Fig. 5. Traffic Classification using ESOM

H. Monitoring Mode

The monitoring mode of T2 produces a RRD database which can be queried using a provided AWK script. The script generates a graph which is updated at a chosen interval, e.g., every second. What makes this mode really powerful is that the features which can be monitored can easily be configured by the user. For example, the number of Jumbo Frames packets (ethertype 0x8870), the number of L2TP (IP protocol 115) or the number of packets sent over TCP port 22 or UDP port 53 can all be monitored. By default, standard values such as number of flows, bytes, packets or alarms are output. Simultaneous monitoring of two fields, such as the number of DNS queries and responses is also possible. Fig. 6 illustrates the monitoring of the number of flows. The black line represent the data forecast by Holt-Winters [31].

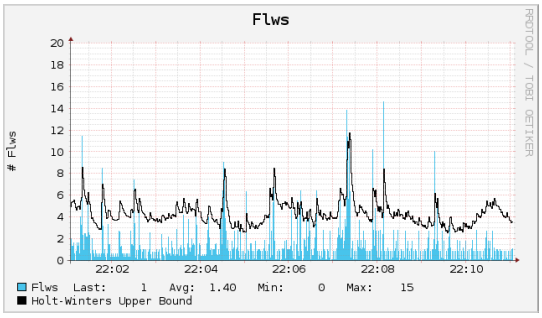

Fig. 6. Monitoring Flows

Thanks to the continuous tab separated stream of data, additional monitoring or forecasting methods can be easily tested. For example, in order to build an improved anomaly detection algorithm based on Holt-Winters, a simple phased locked loop (PLL) can be added to render the system more resilient against disturbances.

I. Forensic Packet Mode

The packet mode of T2 provides a column based packet view, where each packet is linked to a flow by a unique flow index. Hence, an easy drill down from flow to packets and back is possible. By selecting a specific flow index, the Wireshark feature *Follow TCP stream* can be easily reproduced. The *findexer* plugin was specially developed for fast extraction of flows from large pcap files. Thus, malware operations can be detected by analysing the flow output with simple measures such as bit fields or connection oriented anomalies and then isolated in a new pcap. In the packet mode, several flow-based features such as anomaly bitfields can be followed over time and a flow sequence analysis can be extracted from the packet file.

J. Forensic Content Extraction

The key objective in traffic troubleshooting and forensics on large datasets is to use the fastest and most efficient way to locate the problem and to establish a timeline. Thus, flows of different protocols such as UDP and ICMP can be automatically linked. Using global summaries first, then the aggregated flow representation, flows of interest can be isolated. After extracting these flows, a forensic analyst might be interested in inspecting the payload. Although encryption is widely used, unencrypted traffic, such as HTTP or POP3 are still commonplace. T2 can extract all content including multimedia data, such as pictures, videos or attachments, even when packets are out of sequence. The files are stored under user defined directories and named according to the original filename, the flow index, the direction and the packet number in the flow. This naming convention allows the analyst to instantly map the extracted data to the flow and back to the packet.

In the following section, the performance of T2 processing and content extraction is evaluated against Bro, TShark and Wireshark.

V. Performance Evaluation

To evaluate T2 0.6.6 performance, some equally capable open source tools were tested on the same data and under similar configurations. Wireshark [8], TShark and Bro [14] are established solutions for network traffic analysis and as such were selected. Many tools, such as Justniffer [32] and tcpxtract [33], were quickly dismissed, as they could not cope with all the necessary protocol encapsulations.

Initially two pcaps of 44 and 41 GB respectively were chosen. The first one was recorded with reduced snap length in a corporate network and covers a period of an hour. The second one was captured with full snap length by a network operator. It only covers about 3 min and 40s, but has more IPs/s. Both pcaps contain a variety of layer 4 protocols and layer 2 encapsulations, including VLAN and MPLS, plus several cleartext and encrypted L7 protocols. The operator pcap also contains scans, routing anomalies, snapped, mangled and resequenced packets, which are good indicators of the resilience of a tool.

Table I provides more detailed information about the files.

Origin	Duration[s]	Size[GB]	Gb/s	Act IPs	SnapLen
Corporate	3662	44	0.7	1038546	80
Operator	227	41	1.5	177448	1800

TABLE I
PCAP STATISTICS

Wireshark, TShark, Bro and T2 were installed on the same machine with an Intel i7-4930K CPU, Linux Arch 4.2.5-1 and 64GB of memory. Tests on the same HW running Linux Ubuntu 15.10 revealed similar results. The wall-clock time was measured by using the Unix command `time`. Wireshark and TShark were reconfigured to be more efficient, e.g., all automated resolution was switched off. Bro was configured to load the same amount of protocol dissectors as T2. All tests were conducted 10 times and the average runtime was selected.

Bro processed the operator pcap in 424s while T2 only used half of that time (216s). Wireshark could not cope with this amount of data. The time required to complete the corporate pcap was 68 min for Bro and a factor of two faster (32 min) for T2. Bro and T2 proved resilient and were able to process both files, despite the reduced snap length and packet anomalies.

In order to compare T2 with Wireshark and TShark, a 1.3GB subset was extracted from the operator pcap. Although only covering a duration of 6s, it contains more than 21000 active unique source IPs, 400 IP pairs at an average and maximal rate of 1.5Gb/s and 5Gb/s, respectively, with 74608 flows.

Wireshark loaded and processed the pcap file in 20s, while the extraction of multimedia and other content took more than 2 hours and 45 minutes. Moreover, 100% CPU was used until the save button was pressed, resulting in an error message stating that not all pictures could be saved. This error might be intended to report pictures which are not complete, e.g., missing packets. TShark needed more than 35 minutes to extract the same data. Bro was faster and processed the file in 17s. Extracting the content required and additional 12.2s, for a total of 26.2s. T2 loaded and extracted all content with 29 protocol dissection and statistical plugins in 11.8s. Included in this time is the generation and simultaneous write of flow and global statistical and mining files including all HTTP and RTP VoIP.

Finally, Table II summarises the loading and processing time of the different tools and cases.

VI. Conclusions

In this paper, Tranalyzer 2, a high-volume network traffic pre-processor and analyser was presented through a series of real-life scenarios. The tool supplies flow and packet based output, both of which are easy to parse and interface with various databases and SIEMs. Standards such as Splunk, IBM QRadar and IPFIX/NetFlow are all supported. In addition, several post-processing scripts are provided, facilitating the interaction with existing graphical tools such as RRD, Gnuplot or Graphviz.

Tool	Configuration	Time [s]
T2	all open-source plugins (13)	6.4
T2	29 protocol plugins	10.6
T2	HTTP + SIP/RTP content extraction + 29 protocol plugins	11.8
Bro	without saving content on disk	17.0
Bro	saving content on disk	26.2
Wireshark	load and process	20.0
TShark	HTTP content extraction without RTP	2142.0
Wireshark	HTTP content extraction without RTP	12540.2

TABLE II
RUNTIME PERFORMANCE OF VARIOUS TOOLS ON A 1.3GB PCAP

A large amount of statistical key features emerged from practical experience and facilitate analyses in the area of network and application security and troubleshooting. As a platform for rapid development of new intelligent sensors, T2 also became a useful tool in research, particularly for traffic preprocessing to train AI classifiers for anomaly detection.

Various scenarios in the area of troubleshooting, encrypted TM and content extraction for forensics have been discussed. Those real-life cases have highlighted the ability of T2 to compete with, or even replace, existing tools, such as arpwatch or TShark. Even the well established Wireshark and Bro showed shortcomings which T2 was able to complement or even overcome. Run-time performance tests on real world traffic on COTS HW showed that T2 was faster than Bro, TShark and Wireshark by a factor of two, twenty and thousand, respectively.

In conclusion, T2 has evolved into an extremely powerful and versatile tool, truly the Swiss army knife of network/security analysts, troubleshooters and researchers.

VII. FUTURE WORK

Current directions explored by T2 development team are Wireless-specific plugins, Bluetooth traffic and Telecom protocols, such as SIGTRAN. To complement existing signal processing and AI plugins, neural networks, FFT, cross-correlation and shapelets plugins have to be added. Finally, regular expressions and matrix processing are bottlenecks which slow down the system. Thus, an area of research pursued by Tranalyzer team is the combination of dedicated HW, e.g., graphic cards, and software to speed up the system and address the regime beyond 10 Gb/s.

ACKNOWLEDGMENTS

The open source version of Tranalyzer is maintained and funded by RUAG Schweiz AG – RUAG Defence and the Swiss Armed Forces. In this context, we would like to thank R. Sibilia, for his continuous support and fruitful discussions. We are very grateful to T. Ruehl, A. Davolos, F. Albanese and N. Thalheim for their constant efforts in improving Tranalyzer.

REFERENCES

[1] S. Song, L. Ling, and C. N. Manikopoulo, "Flow-based statistical aggregation schemes for network anomaly detection," in *ICNSC*. IEEE, 2006.

[2] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5137.

[3] Tranalyzer Team, "Tranalyzer home page," Jul. 2016. [Online]. Available: http://tranalyzer.sf.net

[4] S. Burschka, T. Ruehl, and F. Buehlmann, "Tranalyzer – netflow extension," in *Proceedings of the 78th IETF*, Jul. 2010. [Online]. Available: https://www.ietf.org/proceedings/78/slides/NMRG-7.pdf

[5] T. . Libpcap, "Tcpdump/libpcap public repository," Jul. 2016. [Online]. Available: http://www.tcpdump.org

[6] J. R. Borque, "Encrypted traffic mining: Ssh command guessing," Master's thesis, EURECOM, Sophia Antipolis, France, 2007.

[7] B. Dupasquier, S. Burschka, K. McLaughlin, and S. Sezer, "On the privacy of encrypted skype communications," in *GLOBECOM*. IEEE, Dec. 2010.

[8] The Wireshark Foundation, "Wireshark home page," Jul. 2016. [Online]. Available: https://www.wireshark.org

[9] B. Claise, "Cisco systems netflow services export version 9. rfc 3954." Oct. 2004. [Online]. Available: https://www.ietf.org/rfc/rfc3954.txt

[10] IBM, "Ibm security qradar siem," Jul. 2016. [Online]. Available: http://www.ibm.com/software/products/en/qradar-siem

[11] NFDUMP, "Nfdump," Jul. 2016. [Online]. Available: http://nfdump.sf.net

[12] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "Sshcure: a flow-based ssh intrusion detection system," in *Dependable Networks and Services*, ser. Lecture Notes in Computer Science, vol. 7279. Berlin, Germany: Springer Verlag, Jun. 2012.

[13] sFlow, "sflow," Jul. 2016. [Online]. Available: http://www.sflow.org

[14] Bro, "The bro network security monitor," Jul. 2016. [Online]. Available: https://www.bro.org

[15] Telecommunication Networks Group – Politecnico di Torino, "Tstat: Tcp statistic and analysis tool," Jul. 2016. [Online]. Available: http://tstat.polito.it

[16] T. Oetiker, "Rrdtool home page," Jul. 2016. [Online]. Available: http://oss.oetiker.ch/rrdtool

[17] CERT NetSA, "Silk: Cert netsa security suite: Monitoring for large-scale network," Jul. 2016. [Online]. Available: http://tools.netsa.cert.org/silk

[18] M. Roesch, "Snort," Jul. 2016. [Online]. Available: https://snort.org

[19] A. Dainotti, W. de Donato, A. Pescape, and G. Ventre, "Tie: a community-oriented traffic classification platform," in *TMA*, 2009.

[20] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification," *Systems Journal, IEEE*, no. 99, Nov. 2014.

[21] ntop, "ndpi: Open and extensible lgplv3 deep packet inspection library," Jul. 2016. [Online]. Available: http://www.ntop.org/products/deep-packet-inspection/ndpi

[22] H. Jiang and C. Dovrolis, "Passive estimation of tcp round-trip times," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, Jul. 2002.

[23] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, Jul. 1997.

[24] M. Mathis, J. Heffner, and R. Reddy, "Web100: Extended tcp instrumentation for research, education and diagnosis," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, Jul. 2003.

[25] L. N. R. Group, "arpwatch: the ethernet monitor program," Jul. 2016. [Online]. Available: http://ee.lbl.gov

[26] N. Guerin, "Encrypted traffic mining for anomaly detection," Master's thesis, EURECOM, Sophia Antipolis, France, 2005.

[27] D. Piccito, "Traffic mining in ip tunnels," Master's thesis, EURECOM, Sophia Antipolis, France, 2006.

[28] A. Davolos, "A feasibility study of signal processing on ip flows towards traffic classification," Master's thesis, Politecnico di Torino, Italy, 2009.

[29] A. Ultsch, "Self-organizing neural networks for visualization and classification," in *Information and Classification*, ser. Studies in Classification, Data Analysis and Knowledge Organization. Springer Berlin Heidelberg, 1993.

[30] T. Ruehl, "Non agglomerative data preparation for a cluster-oriented visualization with emergent structure maps," Master's thesis, Philipps-Universitaet, Marburg, Germany, 2009.

[31] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proceedings of the 14th USENIX Conference on System Administration*, ser. LISA '00. USENIX Association, 2000.

[32] O. Notelli, "Justniffer: Tcp flow sniffer," Jul. 2016. [Online]. Available: http://justniffer.sf.net

[33] N. Harbour, "tcpxtract," Jul. 2016. [Online]. Available: http://tcpxtract.sf.net