Excel@FIT 2018
http://excel.fit.vutbr.cz

Network Traffic Processing in Distributed Environment

Viliam Letavay*

Abstract
The growth of computer networks and the Internet availability opens new opportunities for cybercrime activities. Security administrators and LEA (Law Enforcement Agency) officers call for powerful tools for high-speed network communication analysis of an enormous amount of traffic. The forensic analysis needs for various cybercrime cases may differ. This paper aims to design a novel approach of real-time network traffic processing up to an application layer in a distributed environment. The research focuses on captured traffic analysis and information extraction of multiple application protocols. The solution has to be configurable, scalable and capable to analyze even incomplete communication.

Keywords: Network forensic analysis — Network traffic processing — Actor model

Supplementary Material: Demonstration Video

*xletav00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The expansion of computer networks and the Internet availability opens new opportunities for cybercrime activities and security incidents associated with network applications. The amount of connected devices grows, and the traffic speed increases. Security administrators and LEA (Law Enforcement Agency) officers call for powerful tools that enables them to extract useful information from network communication. The network forensics that is responsible for capturing, collecting and network data analyzing is getting more important.

In the forensic investigation, the network traffic is continuously captured from multiple sources. The captured data has to be processed and analyzed up to the application layer. For LEA, the interesting information from computer network traffic is primarily hidden in application messages such as instant messaging, emails, voice, RTP, localizable information, documents, pictures, etc. Although, the relevance of extracted artifacts may be different from case to case, all cases require at least some kind of digital evidence. The processing system has to be able to extract this data from the traffic, even if it is corrupted.

The analysis of high-speed traffic is easier to achieve in a distributed environment. I have decided to use *actor model* that is effective, and capable of linear scalability. Scalable properties of *actor model* design for network forensics are promising as the VAST platform [1] shown.

In this work, I intend to design such system as a software solution that is also linearly scalable, and platform independent. In comparison to VAST, I want to perform real-time analysis as well with the focus on information extraction from the application layer.

2. Background & Related Work

Network forensics is a process that identifies, captures and analyzes network traffic [2]. Network forensic techniques are used by several network forensic frameworks [3, 4, 5, 6, 7, 8] and tools. In this research apart of framework, I have examined also open source tools. PyFlag and CapAnalysis are capable of offline analysis, as well as Xplico and NetworkMiner that can also perform real-time analysis. All mentioned tools do not support distributed deployment.

The models for distributed processing [9, 10, 11] are more suitable for real-time network forensic analysis from multiple sources, such as logs and captured communication. The models are based on an agent system, where numerous agents performs collection task. The extracted information is sent to the network forensic server and analyzed on this single node [12] only. The *forensic server* is obviously the bottleneck that has to process all the data.

Elimination of all *single points of failure* in proposed, redesigned architecture should remedy this issue. The *actor model* is one of the attractive solutions that solve these problems elegantly and efficiently. The actor model was firstly introduced in 1973 [13]. It comes with a separate unit called *actor*. Actors execute independently and in parallel. They communicate asynchronously via message passing, and their state is otherwise immutable. Actor's behavior determines how to process the incoming message. Actor system is considered to be capable of linear scalability [1].

3. Problem Statement and Solution

Network forensics, is a tedious work that strictly relies on completeness and precision of all undertaken steps to gain a piece of a puzzle that fits together as a shred of evidence. Considering the current speeds of regular users' home network connection(s), an unabridged analysis would require enormous computation resources. Try to imagine, that each network packet would be analysed by many protocol dissectors with a goal to extract for example an acknowledgment of email delivery. To be able to achieve this goal with optimal computation power, I must revisit currently utilized methods and redesign them to work on in distributed environment which brings new challenges to architecture design, algorithms functionality, data synchronization and so on and so forth.

Let's start with an imaginary demonstration. The math is simple, one computer with 1 Gbps NIC (Network Interface Card) that has a relatively simple task to capture traffic during full line load would be required to write to a disk under the constant speed of 1000Mbps $\approx 125\,\mathrm{MB/s}$. Proposed system has to guarantee that no data are lost during the capture. A suspect can simultaneously download and upload data which means that the monitoring device cannot have only one $1 * 1$ Gbps NIC, but it needs $2 * 1$ Gbps cards, one for uplink, one for downlink. Thus, the required speed of continuous disk writing would be $2 * 125\,\mathrm{MB/s} \approx 250\,\mathrm{MB/s}$. Now, if the requirement is to store the communication for one day, the disk capacity have to be $250 * 60 * 60 * 24 \approx 21.6\,\mathrm{TB}$. This is achievable with commodity hardware, e.g., $2 * 12$ TB drives with RAID 0 or $4 * 12$ TB with RAID 1+0 with assumed write/read speed of $250\,\mathrm{MB/s}$. But what if only one day is not enough? For typical forensic case, capturing period spawns through weeks or months.

Assuming, one day is what I need. I am tasked to retrieve valuable information from the captured communication. I know that I can perform the write operation on drives, but for analysis purposes, I need to read the data as well. Concurrent read and write operations on the same hard-drives slows down both, starting with data loss in capturing phase, resulting in irretrievable information during the analysis. I cannot afford either one. We can argue, that a "simple" solution would be to double the count of drives to create a performance buffer and let operating system to deal with it. But what if the analytical process requires more computation power at one moment and overwhelm resources in-spite of the capturing; it would end up with the data loss. What happens with results of the analysis? It need to store them, and where else than on hard-drives that are under continuous pressure from capturing; it end up with the data loss.

From my previous performance measurements, I know that single computation node is limited and commodity hardware is hardly sufficient to perform all required operations in real-time and over long periods of time. Speed of separation of frames into a conversations which needs a dissection of the network protocols up to the application layer, is roughly 300 Mbps [14, pp. 45-51], which is not sufficient. On the other hand, I am confident that application created and optimized for this singular purpose can do the processing faster and breach the 1 Gbps line speed. Nevertheless, I do not believe that single machine solution is capable of doing overall analysis and extraction of information from the application layer. I have to design the solution to be distributed across multiple machines.

The solution is based on the actor model. Each actor represents an independent processing unit. The communication between actors is managed by messaging. The actor has no shared state; thus all actors work in parallel. If actors run on the same node, the message passing has a little overhead compared to a function call or a loop. However, if actors scale over multiple nodes, messages need to be serialized. The serialization process introduces latency and consumes part of processing power.

4. Architecture Design

Incomplete data provided by unreliable traffic interception can lead to skewed results; some information may be lost, some fabricated by reconstruction process. Keeping these facts in mind, the processing cannot strictly follow RFCs and behave like a *kernel* network stack implementation, but it has to incorporate several heuristics. For example, to fill missing gaps in data, and to consider these fillings during application protocol processing, or never to join multiple frames into a single conversation unless it passes more advanced heuristics and checks. Network forensic tools which I have worked with do mostly respect RFCs and thus may produce misleading results as already shown [15].

I propose a distributed architecture with no single point of failure, composed of commodity hardware that will be capable of linear scalability, and capable of fine resource utilization. See Figure 1 for design details.

At the top level, I have divided the entire process into the two main stages:

Data preprossessing Reconstruction of conversations at the application layer (L7 conversations) from the captured traffic. Each of these conversations holds information about the source and destination endpoints, time stamps and reassembled payloads of exchanged application messages.

Data analysis Identification of application protocols in reconstructed L7 conversations and subsequent use of a proper application protocol dissector to reconstruct application events from given conversations (e.g., visited web pages, sent emails, ...). The output of this stage is a set of forensic artifacts.

First stage, data prepossessing, is executed on set of an independent *Reassembler* nodes. Reconstructed L7 conversations from the stream of captured packets can originate from *PCAP files* or can be captured from *the live network interface*.

In the most common use case, there is only one source stream (i.e., one PCAP file) which I want to analyze. Therefore to utilize all of the *Reassembler* instances, I have to split packets from this stream into a smaller sub-streams, which will be distributed among online *Reassembler* instances. For this split, I can not use a naive method such as *Round Robin*. *Reassembler* nodes operate independently of each other and to fully reconstruct L7 conversations (each can consists of multiple packets), they have to obtain all the pieces of the particular L7 conversation.

Using this naive method, there could occur a situation where half the packets from one L7 conversation will end up in one *Reassembler* node and second half in some other; both nodes would end up with incomplete data, and none of them won't be able to reconstruct the conversation entirely.

Solution to this problem is another type of nodes called *L4 Load Balancer*, which will be positioned in front of the *Reassembler* nodes. They will extract source and destination IP addresses and ports from each packet of the source stream and will use them to decide to which instance of *Reassembler* should forward the packet concerning its context. This way, all packets of a particular L7 conversation will always be forwarded to precisely one *Reassembler* instance. The reconstructed L7 conversation will be then stored in a distributed database, ready to be retrieved in the second stage of the execution.

In the *second stage*, a subset of reconstructed L7 conversations is retrieved from the distributed database (by using manual or automatic selection with specified rules) and delivered to the *Snooper* nodes. They will identify used application protocol and use proper application protocol dissector module to extract data from the L7 conversation. Extracted data will be stored back into the distributed database.

Each instance of a particular node acts as an individual actor in the system, communicating with other actors by message passing. Thanks to this design, I am able to distribute the computation across multiple machines maintaining the linear scalability.

5. Conclusion

In this research, I have proposed the system for distributed real-time forensic network traffic analysis up to the application layer capable of processing communication at high speed. I intend to create a system based on actor model that scales linearly and is hardware independent.

My prototype implementation of the proposed system called NTPAC (Network Traffic Processing & Analysis Cluster) is based on C# actor system library *Akka.NET*. Selection of technologies implementing higher abstractions is essential for fast prototype creation. My preliminary measurements conducted on

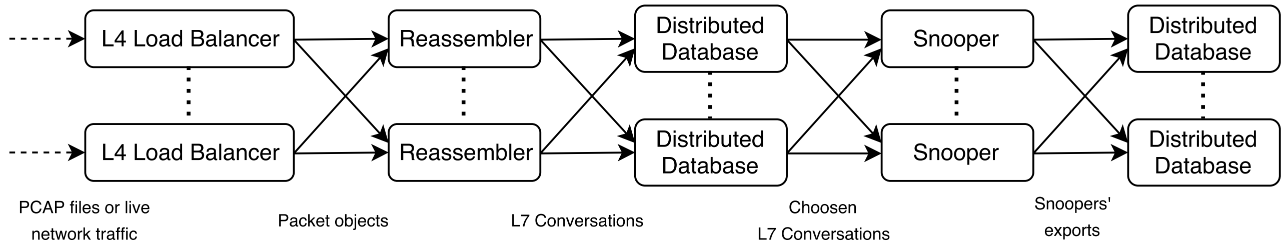

Figure 1. Architecture diagram shows proposed system's nodes with information flow between them. Interconnections between agents are logical, independent of underlying hardware architecture. The solution is to be deployed on a single node or scale up in a distributed environment.

regular workstation[1] show that PCAP files are read and parsed up to application protocol layer with speed of 3929 Mbps on a single core. In comparison, speed capture file loading with a single *Capture* actor is 3482 Mbps/core. Lastly implemented *L4 Load Balancer* operates at speed of 3447 Mbps/2cores. These are very first measurements without more complex performance optimization on an incomplete system. The *Reassembler* supporting TCP and UDP with heuristics [15], *Distributed Database* module, supporting *ArangoDB* or *Cassandra*, are implemented, but not yet integrated.

Acknowledgements

I would like to thank my supervisor Jan Pluskal for his help.

References

[1] Matthias Vallentin. *Scalable Network Forensics*. PhD thesis, UC Berkeley, 2016.

[2] Emmanuel S Pilli, Ramesh C Joshi, and Rajdeep Niyogi. Network forensic frameworks: Survey and research challenges. *digital investigation*, 7(1-2):14–27, 2010.

[3] Slim Rekhis, Jihene Krichene, and Noureddine Boudriga. Digfornet: digital forensic in networking. In *IFIP International Information Security Conference*, pages 637–651. Springer, 2008.

[4] Ahmad Almulhem and Issa Traore. Experience with engineering a network forensics system. In *International Conference on Information Networking*, pages 62–71. Springer, 2005.

[5] Wei Wang and Thomas E Daniels. A graph based approach toward network forensics analysis. *ACM Transactions on Information and System Security (TISSEC)*, 12(1):4, 2008.

[6] Nicole Lang Beebe and Jan Guynes Clark. A hierarchical, objectives-based framework for the digital investigations process. *Digital Investigation*, 2(2):147–167, 2005.

[7] Sundresan Perumal. Digital forensic model based on malaysian investigation process. *International Journal of Computer Science and Network Security*, 9(8):38–44, 2009.

[8] Waleed Halboob, Ramlan Mahmod, Muhammad Abulaish, Haider Abbas, and Kashif Saleem. Data warehousing based computer forensics investigation framework. In *Information Technology-New Generations (ITNG), 2015 12th International Conference on*, pages 163–168. IEEE, 2015.

[9] Wei Ren and Hai Jin. Distributed agent-based real time network intrusion forensics system architecture design. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 1, pages 177–182. IEEE, 2005.

[10] Wei Ren. On a reference model of distributed cooperative network, forensics system. In *iiWAS*, 2004.

[11] Diangang Wang, Tao Li, Sunjun Liu, Jianhua Zhang, and Caiming Liu. Dynamical network forensics based on immune agent. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 3, pages 651–656. IEEE, 2007.

[12] Suleman Khan, Abdullah Gani, Ainuddin Wahid Abdul Wahab, Muhammad Shiraz, and Iftikhar Ahmad. Network forensics: Review, taxonomy, and open challenges. *Journal of Network and Computer Applications*, 66:214–235, 2016.

[13] Carl Hewitt, Peter Bishop, and Richard Steiger. Session 8 formalisms for artificial intelligence a universal modular actor formalism for artificial intelligence. In *Advance Papers of the Conference*, volume 3, page 235. Stanford Research Institute, 1973.

[1] Intel i7-5930K 4.3GHz, 64GB RAM, 512GB SSD

[14] Jan Pluskal. *Framework for Captured Network Communication Processing*. PhD thesis, Diploma thesis, FIT VUT v Brne, 2014, https://www.fit.vutbr.cz/study/DP/DP.php.cs, 2014.

[15] Petr Matoušek, Jan Pluskal, Ondřej Ryšavý, Vladimír Veselý, Martin Kmeť, Filip Karpíšek, and Martin Vymlátil. Advanced techniques for reconstruction of incomplete network data. In *International Conference on Digital Forensics and Cyber Crime*, pages 69–84. Springer, 2015.