Enabling Non-Expert Analysis OF Large Volumes OF Intercepted Network Traffic

Erwin van De Wiel, Mark Scanlon, Nhien-An Le-Khac

Chapter 11

ENABLING NON-EXPERT ANALYSIS OF LARGE VOLUMES OF INTERCEPTED NETWORK TRAFFIC

Erwin van de Wiel, Mark Scanlon and Nhien-An Le-Khac

Abstract Telecommunications wiretaps are commonly used by law enforcement in criminal investigations. While phone-based wiretapping has seen considerable success, the same cannot be said for Internet taps. Large portions of intercepted Internet traffic are often encrypted, making it difficult to obtain useful information. The advent of the Internet of Things further complicates network wiretapping. In fact, the current level of complexity of intercepted network traffic is almost at the point where data cannot be analyzed without the active involvement of experts. Additionally, investigations typically focus on analyzing traffic in chronological order and predominately examine the data content of the intercepted traffic. This approach is overly arduous when the amount of data to be analyzed is very large.

This chapter describes a novel approach for analyzing large amounts of intercepted network traffic based on traffic metadata. The approach significantly reduces the analysis time and provides useful insights and information to non-technical investigators. The approach is evaluated using a large sample of network traffic data.

Keywords: Internet taps, network forensics, traffic metadata analysis

1. Introduction

Lawful interception is used by law enforcement in many middle and high level criminal investigations. Investigations of intercepted traffic cover voice and network data [19]. This work focuses on intercepted network data that is a valuable source of evidence in crimes such as child abuse material distribution [23], cloud hosted services [19], industrial espionage [10], dead drops [10, 17], malicious software distribution [17], instant messaging [22], piracy [18] and illegal content distribution [17].

However, investigating these crimes is challenging for non-technical personnel because of the difficulty in interpreting network data.

Analysis of the data can also be very labor-intensive. Placing a telephone or IP tap requires special investigatory powers. In The Netherlands, the legal power is provided by the Special Investigative Powers Act (Bijzondere Opsporings Bevoegdheid in Dutch). Dutch telecommunications law requires every provider to make data interception equipment available to law enforcement. Law enforcement is also permitted to intercept data at locations without service provider involvement, such as when intercepting data from wireless access points.

Ideally, all digital forensic evidence should be analyzed by expert investigators. However, the reality is that there are simply too many cases that require expert analysis and the case backlog is often too large [15]. As a result, evidence processing by non-experts has become a necessity in law enforcement agencies around the world [16].

Analysis of intercepted data from an IP tap is often conducted by non-technical investigators who analyze the data in chronological order. In a typical scenario, an investigator analyzes data collected over a period of four weeks. The investigator works with the first day of the IP tap data and the analysis software presents all the data for the day starting from 0:00 to 23:59. If the focus is on HTTP-based web traffic, each web visit is displayed in a list view by the software. The investigator then selects a row and the contents of the website are displayed. The only option in such a scenario is for the investigator to examine every web visit sequentially. As a result, it would take a considerable amount of time to analyze the IP tap traffic over the entire four-week period. The analysis software may provide options to filter on text strings and protocols such as HTTP or FTP, but the investigator must know in advance the search terms to be used in the filtering.

A solution is to analyze all the traffic and create text filters based on knowledge about the case. However, this task is challenging because even ordinary Internet users produce vast amounts of traffic, and the traffic will increase significantly as the use of the cloud increases [5, 9].

This chapter proposes a novel approach for analyzing large streams of intercepted data from IP taps. Instead of analyzing the data chronologically, the approach identifies what lies behind an intercepted Internet connection and produces an overview of the extracted information that can be interpreted by non-technical investigators. This chapter also describes the Network Intell system, which instead of focusing on data content, analyzes metadata related to the protocols seen in the data. This reduces the time required to analyze the intercepted traffic. It also provides a quick way to retrieve intelligence information – essentially a

clear view of what is seen behind an IP tap. For example, if an investigator knows that a mobile phone is present, then he can focus on data related to mobile phone protocols. The performance of the system is evaluated using a large sample of network traffic data.

2. Background

Several tools have been developed for analyzing intercepted network traffic. However, the majority of the tools require highly-specialized knowledge for their effective use.

The available tools can be divided in two groups. The first group comprises tools created for professionals with high levels of networking knowledge such as network administrators and digital investigators with advanced network protocol expertise. These tools analyze network traffic at the packet level and extract data for deeper analysis.

The second group comprises specialized network forensic tools that analyze captured traffic. While these tools are not as complex as those in the first group, they still require specialized networking knowledge. Most of the tools are available as freeware or as professional versions that cost between $500 and $1,000. The tools employ the well-known PCAP packet capture format.

2.1 Network Forensic Analysis Tools

The NetworkMiner tool was created by Hjelmvik [7] in 2007. The tool splits network traffic and extracts data from within the traffic. The latest version of NetworkMiner can distinguish between hosts, but this feature cannot be used when a network connection in a lawful interception scenario is analyzed; in such a scenario, the captured traffic appears to come from a single IP address (tapped IP address). Additionally, NetworkMiner does not scale well with increasing data size. For example, the analysis of a 1.25 GB capture takes more than 21 minutes. NetworkMiner parses all the information and the results may be analyzed by scrolling the list view. However, the tool is too advanced to be used effectively by non-technical investigators.

The Xplico tool extracts application data from a network capture [3]. Network traffic is broken down to the protocol level and metadata is extracted for each protocol. Xplico can be used to analyze relatively large network capture datasets. It parses all the information and the results can be viewed and analyzed via a web interface. Although the tool is easy to operate, an investigator still needs to review and analyze the data. Moreover, the tool cannot identify the devices behind the

captured network point because it does not incorporate the knowledge required to interpret the data.

The popular Wireshark network analyzer breaks traffic down to the packet level, and can even analyze broken and half packets. Wireshark also supports the reconstruction of network flows, data analysis using advanced filtering and deep packet inspection. However, Wireshark can be difficult to use by non-technical investigators. Additionally, it is relatively slow when performing filtering and running modules. This is because each filter and module must be executed on the entire dataset. Filter creation is also very slow because each filter has to be tested on the entire database. Wireshark only supports single capture files, although it is possible to connect multiple captures using third-party tools. Moreover, Wireshark requires significant amounts of time to analyze the large volumes of intercepted network traffic encountered in a forensic investigations involving residential or corporate users.

2.2 Traffic Metadata Analysis Tools

Several tools are available for examining and/or extracting metadata from capture files. An example is the `p0f` tool [24]. This tool employs passive traffic fingerprinting to identify the entities involved in TCP/IP communications. However, in order to obtain the best results, the tool must be executed on the target network (i.e., network on which the interception is conducted). This is almost never possible in the case of lawful interception because the actual capture takes place at the Internet provider. The `p0f` tool can only identify or fingerprint traffic coming from the tapped IP address, which is typically a single router or gateway.

Justniffer is another tool that focuses on metadata in network capture files [12]. The tool targets request and response information from various protocols. Justniffer exports information in a log format and provides Python scripts that support network forensic functionality.

3. Network Intell System

This section specifies the requirements of a system for analyzing large amounts of intercepted network traffic. The requirements are realized by the proof-of-concept Network Intell system, which is also described in this section.

3.1 System Requirements

The Network Intell system is not intended to be a one-size-fits-all solution. Instead, it should greatly facilitate the analysis of large amounts of network traffic to discover evidence relevant to digital forensic investiga-

tions. Another key requirement is speed of analysis. Casey [1] has specified a number of requirements for network traffic processing tools [1]. The Network Intell system should satisfy the following requirements:

- **Tcpdump Format:** This format is desired because the WinPcap library is used to parse network traffic. Thus, the proposed system should be designed to handle PCAP traffic.
- **Reliable Protocol Identification:** Deep packet inspection must be used where possible to identify different protocols.
- **Data Reduction:** The actual contents of rebuilt packet streams need not be stored. Instead, the system should focus on metadata and only store packet information. This increases the speed of automated analysis and reduces the storage requirements.
- **Keyword Search:** The system should identify devices and their usage behind an Internet connection based on custom queries. It particular, it should search for metadata items using keywords. Since data content is not stored, it is not possible to search the contents.
- **Read-Only Feature:** The parsed traffic is stored in an SQLite database. The stored data should be accessed as needed, but no modifications should be made to the database.

A key requirement is that Network Intell should be operable by digital investigators and/or non-technical investigators. A digital investigator would be able to input and change the rules that identify the devices and objects behind an intercepted network point. A non-technical investigator would simply operate the system and report the results in an investigative report or pass the results for further analysis by a digital investigator. Any user should be allowed to edit the rules.

3.2 System Design

Figure 1 shows the Network Intell architecture. The system has three main components: (i) network parser; (ii) fragmented traffic reassembler; and (iii) protocol analyzer/parser.

Network Parser. The system splits protocol data to produce usable metadata. This metadata is stored in a SQLite database that may be queried later for devices and objects behind the intercepted network address. The technical requirements define the traffic that is used as a

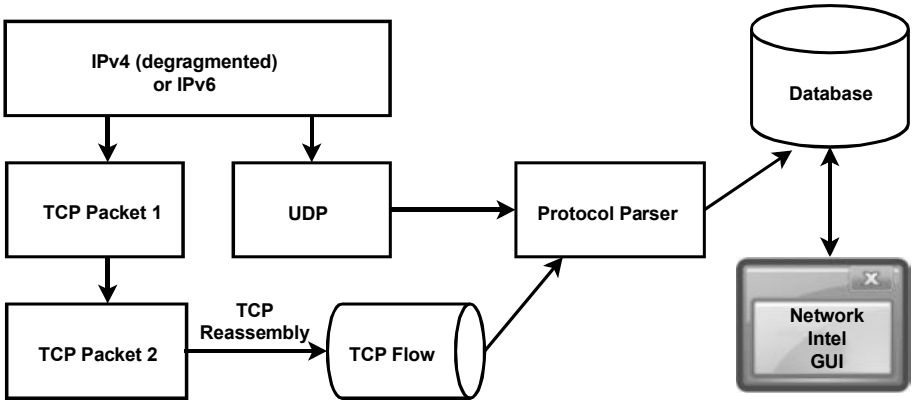

Figure 1. Network Intell architecture.

parsing object. The most useful protocols in the traffic are selected for parsing. Each parsed protocol is stored in its own database table.

WinPcap provides options for intercepting live network traffic and opening stored packet captures (PCAP files). However, the Network Intell system is designed to only operate on stored packet captures. As stated above, the interception of network packets is done at a service provider. The function `pcap_open_offline` in WinPcap is used to open offline stored packet captures. By opening a packet capture in this manner, it is possible to loop through all the stored packets.

The first requirement is to only include Ethernet packets from the network access layer. The next requirement is to only parse Ethernet packets with IPv4 or IPv6 headers from the Internet layer. This is motivated by the fact that, as of June 2016, 10.88% of the computing systems that accessed Google used IPv6 [11] and the percentage has grown to 18.09% as of January 2018 [6]. The parser selects Ethernet packets with IPv4 or IPv6 headers associated with the TCP, UDP or ICMP protocols. At this stage, ICMP packets are also logged.

The final requirement is the application layer. This layer is the most important because the final results stored in the database are associated with the application layer protocols that are selected.

Fragmented Traffic Reassembler. The most complex part of network analysis is packet reassembly. Packet reassembly occurs at two layers: (i) IP layer; and (ii) TCP layer.

IP fragmentation can be caused by IPv4 routers that fragment IPv4 packets when they are transferred to other networks [13]. Reassembly of the fragmented packets is done by the receiving endpoint. Since IPv4

packets do not necessarily arrive in order, it is difficult to reassemble the packets in the correct order without errors. Fragmented packet reassembly is performed using the IP datagram reassembly algorithms described in RFC 815 [2].

Network Intell uses code implemented in the IPTraf tool [8]. Several attack vectors target the manner in which IP fragmentation is handled. However, protection from IP fragmentation attacks is beyond the scope of this work. In any case, IPv4 fragmentation is not seen very often and IPv6 routers do not support fragmented IPv6 packets.

TCP reassembly is the next step in rebuilding network streams and data. Analysis tools rely on proper TCP reassembly to rebuild traffic.

Rebuilding TCP packets produces streams (also called flows). Considerable research has focused on TCP reassembly (see, e.g., Wagener et al. [21]). Problems that occur during IPv4 reassembly also occur in TCP reassembly. Exploits can be used to bypass the reassembly process and can even crash the software. Protection against TCP reassembly attacks is not in the scope of this research, although it could be considered in future tests of the system.

Network Intell uses a portion of the code in the `tcpick` TCP reassembly tool [4]. The modified code also incorporates a requirement that every TCP connection must have a complete three-way handshake. The three-way TCP handshake involves an exchange of packets before a TCP connection is established.

In a three-way TCP handshake, three packets are exchanged between a client and server with the first SYN packet being the most important packet. ACK and SEQ numbers are used in the reassembly process. The sender of the SYN packet is the initiator of the connection and is identified as the "TCP flow from" entity. Without a SYN packet it would be very complex to identify the entity that started the connection because packets may not arrive in the same order as they were sent. Therefore, a complete three-way handshake is needed before a connection can be parsed for a higher-level protocol.

Network Intell relies on a four-tuple mechanism for packet reassembly: (i) source IP address; (ii) destination IP address; (iii) source port; and (iv) destination port. Each reassembled connection must start with a SYN; this is needed to identify the entity that started the connection and, thus, determine the client side and the server side. It is possible to create flows that do not have starting SYNs, but this introduces packet reassembly errors. Specifically, the sending and receiving endpoints can get switched and incorrect assumptions are made during traffic analysis. An example is when an HTTP request from an external entity is directed to the intercepted network point. Without a SYN packet, the request

could be identified as traffic originating from inside the intercepted network point and headed outside the network.

Network Intell rebuilds each flow in memory and stores the metadata associated with the flow in the database. The stored metadata includes the client and server IP addresses, the duration of the flow and the amount of traffic that was intercepted. Flows remain active until a RST or FIN packet is received in the TCP flow. After the flow information metadata is exported to the database, the flow is removed from memory.

Network Intell also provides the option to export a complete flow. This is useful for analyzing specific connections and data transfers. However, the option is currently disabled because disk I/O slows down the analysis. When a flow does not receive the RST or FIN terminating packets, then the flow needs to be shut down after a period of time. This feature is built into Network Intell, but more testing is required to determine the correct timing for flow termination. In any case, since Network Intell works on captured traffic, real-time analysis of live traffic is not an issue.

Protocol Analyzer/Parser. After IP packets are defragmented and TCP packets are reassembled to create a stream or flow, they are parsed by the protocol analyzer. The first step is to identify the contents of the network stream. Casey [1] recommends that traffic should not be filtered based on protocol because of the risk that other traffic can be tunneled through a protocol such as HTTP. For the same reason, it is not advisable to filter traffic based on port number. Although port numbers are exported to a log file, the actual contents of the reassembled packets are used to identify the protocols employed in the application layer. The only exception is the DNS protocol, which is analyzed based its port number (port 53).

Filtering the contents of TCP streams is referred to as deep packet inspection. This technique is useful for detecting protocols hidden inside network traffic. However, it is a complex process because it can cause false hits based on the keywords used to identify protocols. Therefore, it has to be continually evaluated and adjusted where needed, a task that can only be done by a skilled network forensic investigator.

The key functions of Network Intell are to analyze different application layer protocols and store important metadata in a SQLite database. Network Intell conducts deep packet inspection by searching for specific keywords in packet headers. First, it looks for protocol-specific details such as an HTTP header that ends with "\r\n\r\n." Next, it searches for specific keywords such as "GET" and "POST" that are bound to the protocol. Based on the results, the reconstructed packets are identified by their protocol. This method is based on the approach used by Xplico

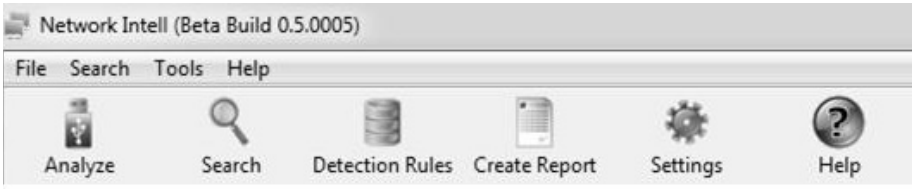

Figure 2. Network Intell main menu.

to identify protocols. Currently, Network Intell employs deep packet inspection to identify HTTP, FTP, SMTP and POP3 protocol packets. Because deep packet inspection is used, these protocols are detected even when they run on ports other than their default ports.

4. System Implementation

As described above, Network Intell analyzes intercepted network traffic. Win32 C was chosen as the programming language to ensure adequate speed of analysis. The Pelles C Microsoft Windows development environment was used to create 32/64-bit C language code [14].

Network Intell maintains several logs. The results of each analysis and the network packets can be exported to a separate log file. The system also can export each TCP stream/flow to a binary file. This enables research to be conducted on unknown network traffic and the content to be verified for consistency. Figure 2 shows the Network Intell main menu functionality, which includes analysis, search, detection rule editing, report creation, settings and help.

The main – and initial – function is analysis. A non-technical investigator can execute this function if he/she knows which capture files need to be investigated. The system asks for the names of the network capture files in the PCAP format. After the files are input, the MD5 and SHA1 hashes of the files are computed. Also, the numbers of packets in the files are counted. The PCAP format does not save packet counts in the header. The only safe way to count packets is to open a capture file and loop through all the packets with a counter running. The result of the analysis is a tree view with an overview of the analyzed items. Since all the metadata is stored in a SQLite database, a user is allowed to include new items as shown in Figure 3.

Detection rule editing is another important Network Intell function. The detection rule editor facilitates the input and modification of rules. A detection rule type must be selected, the result name entered, the parent selected and, of course, the search query entered with partial or exact match options (Figure 4).

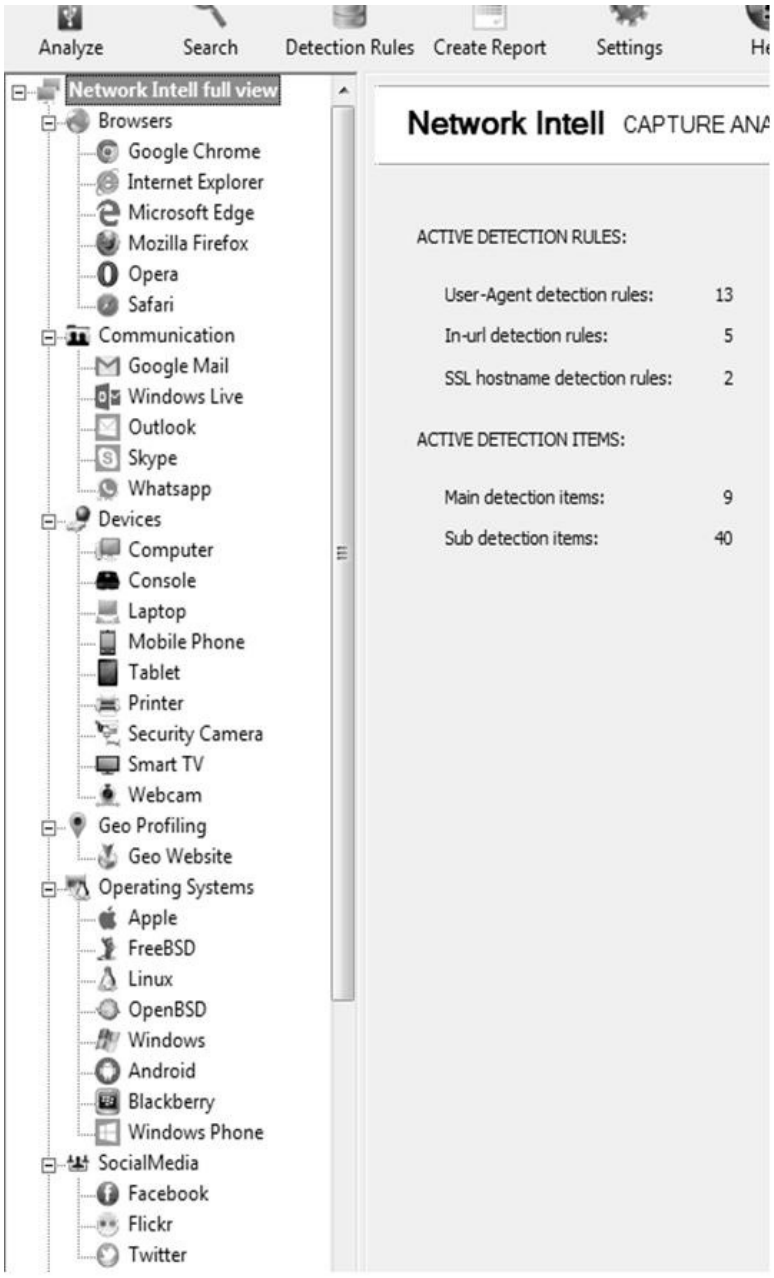

Figure 3. Network Intell analysis options.

5. System Evaluation

This section describes the results of evaluating the Network Intell system. The evaluation used intercepted network traffic in open source captures and home network captures.

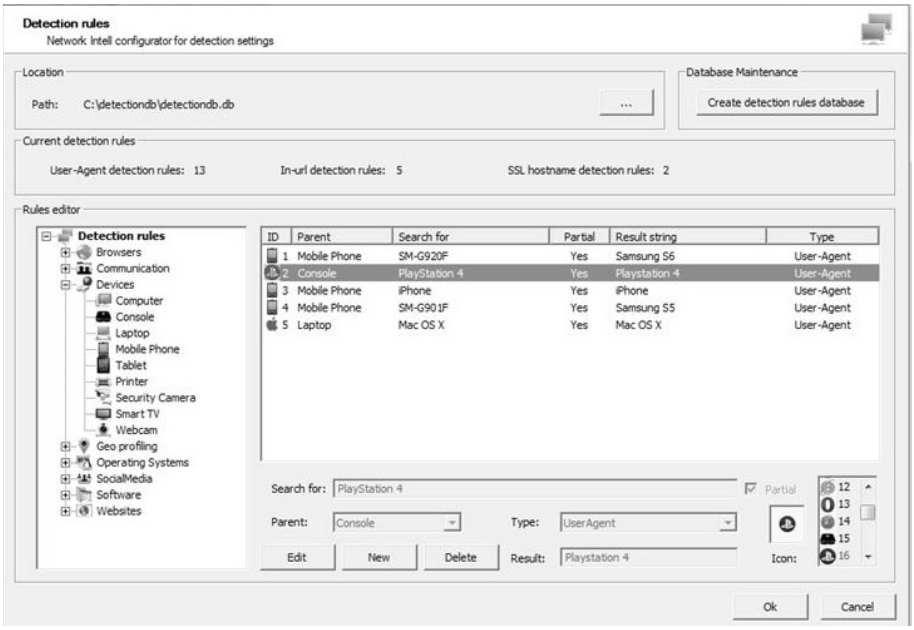

Figure 4. Network Intell detection rules.

The following evaluation scheme was employed:

- Each protocol was tested using a protocol-specific PCAP file.
- The results obtained using Network Intell were compared against those obtained using other network forensic tools such as Wireshark. The only differences were seen at TCP stream/flow level. Wireshark also identifies streams without a starting SYN packet; however, Network Intell cannot handle such streams and needs a SYN packet to identify a TCP stream/flow.
- Personal network traffic was captured at the router level to mimic intercepted network traffic. Most errors were detected during this test because personal network traffic has a lot of content that is not handled by Network Intell. An example is secure DNS traffic, which causes an error in the parsing code.
- Network Intell has not as yet been tested on a real-world investigation. The system is designed to be used by experts as well as non-experts. However, it will be extended to enhance usability; complex configurations will not be required to use the final system.

Performance tests were conducted to investigate how long it would take to analyze an average network tap. The results revealed that Network Intell can analyze and export about 1 GB per minute depending on the computing equipment that hosts the system. This measurement does not consider the exporting of every TCP stream/flow; this would increase the analysis time considerably because a new file has to be created for every TCP stream/flow. Network Intell was instrumented with timers to compute the overall execution time. Considerable disk activity was observed when the execution data was analyzed. This was caused by the extensive logging for error checking and program flow analysis; the execution time was also increased significantly.

As a result, the first measurements were taken with logging enabled. In the first run, 1 GB of test data was used with logging enabled; the time required was measured as 63 seconds on a standard workstation. In the second run, the logging options were disabled and the same 1 GB of test data was processed in just 35 seconds.

Network Intell produces a database of protocol-specific information that can be analyzed further using a SQLite viewer. A query against the SQLite capture database runs almost instantly. The only limitation is when the amount of data approaches the maximum size of a SQLite database (more than 140 TB). No performance testing was performed on the analysis time because it takes less than a second to produce the tree view. The information presented after the analysis of a PCAP file is completed can be used by a digital forensic investigator to check the number of streams that were analyzed and for any errors that may have occurred.

The research goal has been to develop a system that produces useful results from large volumes of data intercepted via a network tap. Instead of analyzing the data in a chronological manner, the Network Intell system identifies what lies behind the intercepted connection and produces an overview of the discovered information. The system is easy to use by experts as well as non-experts; in fact, it can be operated almost via single button actions. Network Intell is fast – it analyzes approximately 1 GB of data per minute with logging enabled and requires just 34 seconds to analyze 1 GB of data without logging enabled. This analysis speed is comparable to or exceeds the performance of other available tools. Finally, Network Intell appears to be able to handle "big data" scenarios. Specifically, good performance was achieved with 1 GB capture files; in contrast, other network analysis tools either cannot handle files of this size or become very slow after loading files of this size.

6. Conclusions

The chapter has presented a new approach for analyzing large amounts of intercepted network traffic and Network Intell, its proof-of-concept implementation. Network Intell exhibits good performance – it can analyze intercepted network traffic in the PCAP format at speeds of around 20 seconds per 200 MB. The analyzed information is stored in a SQLite database that can be queried to identify the devices used behind a network tap and obtain useful statistics about the captured traffic. Network Intell can be used by digital forensic investigators with technical expertise as well as by non-expert investigators.

Network Intell offers a good mix of functionality and performance, but is a proof-of-concept system and several enhancements are possible. For example, the current version engages detection rules input by users. Future research will attempt to employ the Fingerbank database [20] accessible at `www.fingerbank.org`, a device database containing MAC addresses and user-agent device detection information. Additionally, it is ineffective to employ complete user agent strings in lawful interception scenarios because browser software is updated very frequently. Consequently, future work will use detection rules based on browser names instead of complete user agent strings with version numbers.

References

[1] E. Casey, Network traffic as a source of evidence: Tool strengths, weaknesses and future needs, *Digital Investigation*, vol. 1(1), pp. 28–43, 2004.

[2] D. Clark, IP Datagram Reassembly Algorithms, RFC 815 (`tools.ietf.org/html/rfc815`), 1982.

[3] G. Costa and A. De Franceschi, Xplico: Open Source Network Forensic Analysis Tool (NFAT) (`www.xplico.org`), 2018.

[4] duskdriud, `tcpick` version 0.2.1 (`tcpick.sourceforge.net`), 2005.

[5] J. Farina, M. Scanlon, N. Le-Khac and M. Kechadi, Overview of the forensic investigation of cloud services, *Proceedings of the Tenth International Conference on Availability, Reliability and Security*, pp. 556–565, 2015.

[6] Google, IPv6 Adoption Statistics, Mountain View, California (`www.google.com/intl/en/ipv6/statistics.html`), 2018.

[7] E. Hjelmvik, Passive network security analysis with NetworkMiner, *(IN)SECURE Magazine*, issue 18, pp. 18–21, October 2008.

[8] G. Java, IPTraf: IP Network Monitoring Software (`iptraf.seul.org`), 2005.

[9] T. Lillard, *Digital Forensics for Network, Internet and Cloud Computing: A Forensic Evidence Guide for Moving Targets and Data*, Syngress, Burlington, Massachusetts, 2010.

[10] B. Nelson, A. Phillips and C. Steuart, *Guide to Computer Forensics and Investigations*, Cengage Learning, Boston, Massachusetts, 2016.

[11] V. Nicolls, N. Le-Khac, L. Chen and M. Scanlon, IPv6 security and forensics, *Proceedings of the Sixth International Conference on Innovative Computing Technology*, pp. 743–748, 2016.

[12] O. Notelli, Justniffer, Plecno, Milan, Italy (`justniffer.sourceforge.net`), 2014.

[13] N. Olifer and V. Olifer, *Computer Networks: Principles, Technologies and Protocols for Network Design*, John Wiley and Sons, Chichester, United Kingdom, 2006.

[14] P. Orinius, Pelles C (`www.smorgasbordet.com/pellesc`), 2017.

[15] D. Quick and K. Choo, Impacts of increasing volume of digital forensic data: A survey and future research challenges, *Digital Investigation*, vol. 11(4), pp. 273–294, 2014.

[16] M. Scanlon, Battling the digital forensic backlog through data deduplication, *Proceedings of the Sixth International Conference on Innovative Computing Technology*, pp. 10–14, 2016.

[17] M. Scanlon, J. Farina and M. Kechadi, Network investigation methodology for BitTorrent Sync: A peer-to-peer based file synchronization service, *Computers and Security*, vol. 54, pp. 27–43, 2015.

[18] M. Scanlon, A. Hannaway and M. Kechadi, A week in the life of the most popular BitTorrent swarms, *Proceedings of the Fifth Annual Symposium on Information Assurance*, pp. 32–36, 2010.

[19] H. Schut, M. Scanlon, J. Farina and N. Le-Khac, Towards the forensic identification and investigation of cloud hosted servers through non-invasive wiretaps, *Proceedings of the Tenth International Conference on Availability, Reliability and Security*, pp. 249–257, 2015.

[20] J. Spooren, D. Preuveneers and W. Joosen, Mobile device fingerprinting considered harmful for risk-based authentication, *Proceedings of the Eighth European Workshop on System Security*, article no. 6, 2015.

[21] G. Wagener, A. Dulaunoy and T. Engel, Towards an estimation of the accuracy of TCP reassembly in network forensics, *Proceedings of the Second International Conference on Future Generation Communications and Networking*, vol. 2, pp. 273–278, 2008.

[22] D. Walnycky, I. Baggili, A. Marrington, J. Moore and F. Breitinger, Network and device forensic analysis of Android social-messaging applications, *Digital Investigation*, vol. 14(S1), pp. S77–S84, 2015.

[23] A. Yasinsac and Y. Manzano, Policies to enhance computer and network forensics, *Proceedings of the IEEE Workshop on Information Assurance and Security*, pp. 289–295, 2001.

[24] M. Zalewski, `p0f` (`lcamtuf.coredump.cx/p0f3`), 2014.