A transparent bridge for forensic sound network traffic data acquisition

Stefan Kiltz, Mario Hildebrandt, Robert Altschaffel, Jana Dittmann
Otto-von-Guericke University Magdeburg, Germany
Research Group on Multimedia and Security
{kiltz, dittmann}@iti.cs.uni-magdeburg.de, {mhildebr, altschaf}@cs.uni-magdeburg.de

Abstract: In this paper we introduce a prototype that is designed to produce forensic sound network data recordings using inexpensive hard- and software, the Linux Forensic Transparent Bridge (LFTB). It supports the investigation of the network communication parameters and the investigation of the payload of network data. The basis for the LFTB is a self-developed model of the forensic process which also addresses forensically relevant data types and considerations for the design of forensic software using software engineering techniques. LFTB gathers forensic evidence to support cases such as malfunctioning hard- and software and for investigating malicious activity. In the latter application the stealthy design of the proposed device is beneficial. Experiments as part of a first evaluation show its usability in a support case and a malicious activity scenario. Effects to latency and throughput were tested and limitations for packet recording analysed. A live monitoring scheme warning about potential packet loss endangering evidence has been implemented.

Keywords: IT-Forensics, Network security, Reactive security, Intrusion detection

1 Introduction

Computer networks are subject to failure both for unintended and malicious reasons. Such failures are usually attended by incident responders. However, in some cases it is not only important to identify and address the root cause of a malfunction or compromisation of computer networks. Depending on the type of the computer network together with the costs of its downtime, a detailed forensic analysis may become necessary (e.g. to determine which data has been compromised). A forensic model can be used in order to ensure that sound forensic principles are applied throughout the whole investigation. The focus within this paper is the description of a forensic sound network data collection procedure by employing a self-developed tool named "Linux Forensic Transparent Bridge (LFTB)". It is designed to run on inexpensive and readily available hardware and with no costs for software, clearly distinguishing it from costly solutions as provided by e.g. Narus [Nar10]. In the following section the underlying theoretical foundation of our model of the forensic process is introduced. After that, design concepts and their implementation of the LFTB that collects network data with respect to *integrity* and *authenticity* as well as *confidentiality* and *privacy* is shown in detail. Also issues of the deployment of such a device are discussed. Two application examples as part of a first experimental proof of the utility and scalability are shown. The throughput and the latency were tested, addressing the issues of scalability and potential packet loss when employing the LFTB in the field.

2 A holistic model for the forensic process

To gather network data in a forensically sound manner, the security aspects of *integrity*, *authenticity*, *confidentiality* and *privacy* need to be observed (see also [Bis06]). The latter

is relevant because a collection of network data is bound to include data that is protected by data privacy regulations. Different models already exist to describe the forensic process (see for instance [Fre07] and [New07]). Our model extends the one presented in [Fre07] and consists of three main aspects:

- The grouping of investigation steps that belong together logically into **phases**.
- The classification of forensic methods into **classes of forensic methods**.
- The classification of forensically relevant **data types**.

The introduced model covers support cases or hardware/software failures and malicious intent. The three aspects of the model are covered in the following subsections.

2.1 Phases of the forensic process

In our model, six mutually exclusive phases of the forensic process could be identified:

- **Strategic preparation** SP, i.e. measures taken prior to an incident.
- **Operational preparation** OP, i.e measures of preparation for a forensic investigation after a suspected incident.
- **Data gathering** DG, i.e. measures to secure evidence.
- **Data investigation** DI, i.e. measures to evaluate and extract data for further investigation.
- **Data analysis** DA, i.e. measures for detailled analysis and correlation between digital evidence.
- **Documentation** DO, i.e. measures for the detailed documentation of the whole investigation.

The phases of the forensic process do not always follow the chronological order in which they are listed, e.g. the findings during the data investigation phase can lead to subsequent data gathering phases on the same or a different computer system.
During the conduct of each phase, a *process accompanying documentation* should be implemented, i.e. every step taken has to be extensively documented. The final phase of a forensic investigation is the *final documentation*, i.e. the processing of all parts of the process accompanying documentation and the destillation into a report varying in technical detail depending on the audience.
With the Linux Forensic Transparent Bridge (LFTB), the phase of **strategic preparation** SP is of a considerable importance (see also Section 5). Only by looking at IT-forensics from the perspective of the operator of an IT-system, anticipating system failures and malicious activities and therefore practise strategic preparation, the usage of our proposed LFTB becomes useful.
The LFTB can gather data that is relevant to data protection and privacy legislation. The operator of the IT-system has to make sure that he conforms with the legal requirements ahead of any incident.
The decision, what amount of data is to be collected and from which particular LFTB (if multiple instances accross the network exist) form part of the **operational preparation** OP as they rely heavily on the symptom and other incident dependent factors.

2.2 Classes of forensic methods

In our model, forensic methods are not exclusively represented by dedicated forensic toolkits. A lot of software installed on computer systems has forensic capabilities (e.g. logging facilities, anomaly detection). In our model we categorise them according to their type in six classes of forensic methods:

- **Operating system** OS. This class contains methods provided by the operating system (see also [BW06]).
- **File system** FS. This class contains methods provided by the file system (see also [BW06]).
- **Explicit means of intrusion detection** $EMID$. The characteristic of EMID methods provided by extra software is that they are executed autonomously on a routine basis.
- **IT-Application** ITA. This class contains methods provided by IT-Applications run by the user. In addition to their main functionality they also provide forensic methods.
- **Scaling of methods for evidence gathering** SMG. This class contains methods to further collect evidence unsuited for routine usage in a production environment (e.g. false positives, high CPU demands etc.). The LFTB is such an example. It would be undesirable to routinely having the LFTB collect the whole network traffic because of the resulting high data volume.
- **Data processing and evaluation** DPE. This class contains methods which support a detailed forensic investigation, display, processing and documentation.

Those classes of methods are mutually exclusive.

2.3 Data types

Data types are important to describe forensic relevant data of the target computer system, which the forensic tools use as input. The idea is to model forensic data types within a computer system with a layered approach similar to the widely known ISO/OSI network layer model [Zim80]. Although at present our model of data types introduced in the following is considered to cover forensically relevant data within a computer system, it can be extended by new data types in principle. Currently eight data types are suggested:

- **Hardware data** DT_1. Hardware data is data in a system, which is not or only in a limited way influenced by the operating system and applications. Examples are RTC-clock, serial numbers of hardware devices or the code of firmware of hardware devices. This includes virtualisation data, these are easily changed by the host OS but not by the client OS.
- **Raw data** DT_2. Raw data is a sequence of bits (or data streams) of components of the system not (yet) classified. In principle they can contain data of all the other data types. Examples of raw data are memory- and mass-storage dumps. Network packets also constitute raw data.
- **Details about data** DT_3. Details about data constitute meta data added to user data. These can be stored within user data or externally. Details about data can be persistent or volatile. Examples are MAC-times of files or sequence numbers of network packets.
- **Configuration data** DT_4. Configuration data is data that can be changed by the OS or applications, which modify the behaviour of the system, but not its behaviour with regards to communication. That includes the configuration of hardware, of the OS and applications.
- **Communication protocol data** DT_5. Communication protocol data comprises data, which controls the behaviour of the system with regards to communication. That includes, amongst network configuration data, also inter-process communication (e.g. pipes and RPC) of IT-applications.

- **Process data** DT_6. Process data is data about a running process. That includes, for example, the status of the process, the owner of the process, its priority, memory usage or the related application. In IT-applications these can be single threads or data about them.
- **Session data** DT_7. Session data constitutes data collected by a system during a session, regardless of whether the session was initiated by a user, an application or the OS. This includes, for example, started programmes, visited websites or documents within a user session.
- **User data** DT_8. User data are contents created, edited or consumed by the user. This includes, for example, media data such as pictures, texts, audio or video data.

The data represented in the layers, however, is **not** mutually exclusive; the same data that constitutes raw data can also form user data, the difference is the semantics, in which the data is looked at. Although during the usage of the LFTB, in principle, almost all data types can be collected off the raw data DT_2, the main focus is on the Communication protocol data DT_5 and User data DT_8. In the following section we describe the features of the LFTB in detail.

3 Designing forensic software using software engineering techniques

In [Som06] three generic software process models are described. For our project we chose a mixture of the waterfall model and component-based software engineering, since we start from scratch with the design of the LFTB but use parts of the functionality already existing from other authors. For a generic forensic application the requirements to be analysed are methods for $integrity$ and $authenticity$ protection. Depending on the processed data the security aspects of $confidentiality$ and $privacy$ need to be included. Common for all forensic applications is the demand for documentation. Within our model for the forensic process this is the process accompanying documentation (see Section2.1). The resulting software design is shown in Figure 1.

Figure 1: Software design for a generic forensic application

All requirements are incorporated in our LFTB. The process accompanying documentation is a central component for the forensic application, sometimes needing protection for privacy and confidentiality reasons. The implementation and integration phases include the testing of units and the final system (see Section 7).

4 The concept, implementation and usage of a Linux Forensic Transparent Bridge (LFTB)

We designed the Linux Forensic Transparent Bridge to be a universal tool for capturing network data (raw data DT_2) in a sound forensic manner to store evidence on a mass storage device. It is built around the following concepts:

- Integrity of the captured network data
- Integrity of the reports
- Authenticity of the reports and the captured network data

- Integrity of the operating environment
- (Optional) confidentiality of the captured data to preserve privacy
- Stealthy operation within the network

Most of these requirements meet those defined for generic forensic applications. It is of the highest importance, that the source data is not altered by the LFTB during the capturing and storage process. Hence, a cryptographic hash sum is calculated to prove the $integrity$ of the data. The LFTB calculates a cryptographic hash sum using the sha256 algorithm during the capture operation[1]. The LFTB meets the requirement of a accompanying documentation by keeping a detailed log and also ensuring the integrity of that documentation by calculating a cryptographic hash sum, again using the sha256 algorithm. The security aspect of $authenticity$ (i.e. establishing a link of the recorded data to the actual incident) is ensured by a sha256-hmac which uses a user selected passphrase and the current hardware configuration (DMI data) of the computer running LFTB. Hence, our solution ties the gathered network data both to the investigator and the LFTB hardware.We select the sha256-hmac over the use of certificates, which are difficult to employ when using a read-only medium; those certificates would have to be supplied externally.
Instead of developing all functions from scratch the LFTB uses commonly accepted applications and extends them to meet the requirements for forensic applications (see Figure 2).

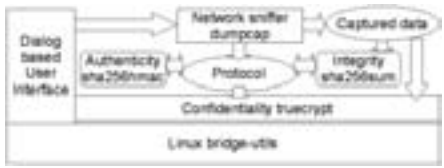

Figure 2: LFTB schematic

The LFTB is a bootable CD-ROM image and comes together with a sha256 publicly comparable cryptographic hash sum, thus ensuring the $integrity$ of the executed code.
We designed the LFTB to be particularly stealthy, which is necessary when recording intended malicious incidents. However, this can only be achieved if the LFTB is deployed in bridging mode or with network taps, since the presence of monitor ports might be detectable via SNMP. Optionally also means of keeping the security aspect of $confidentiality$ and $privacy$ are available. Making this feature optional became necessary, if the hardware platform chosen is not powerful enough to perform both the capturing and encryption processes fast enough. It is, however, positively advised to use the option so as not to collide with data privacy regulations.The following Figure 3 summarises the basic concept.

Figure 3: Functionality of the Linux Forensic Transparent Bridge

We designed the LFTB to record data onto a storage media, together with the documentation script and the cryptographic hash sum, for further analysis. The LFTB is to be used

[1] We select the sha256 algorithm over the popular md5 algorithm despite the higher demands on computational power because the latter is not seen as secure enough with regards to intended collisions.

in the phases of $strategic\ preparation(SP)$, $operational\ preparation(OP)$ and during the $data\ gathering(DG)$ phase according to the model introduced in Section 2. It is a $Scaling\ of\ methods\ for\ evidence\ gathering\ SMG$ method and collects raw data content DT_2 during the data gathering phase DG. The LFTB writes the captured data in a structured file in *pcap* format[2] which can contain all of the forensic data types DT_1 to DT_8. That data is then used in the following phase of data investigation DI to mainly extract Communication protocol data DT_5 and User data DT_8.
Data is captured on a conventional, readily available, x86-based computer system with two network interface adapters installed. The computer has to be equipped with an optical drive capable to boot off a CD-ROM. Mass storage devices need to be attached to the computer (preferably external media such as USB disc drives) to record the captured data together with cryptographic checksums and scripts as part of the accompanying documentation. A generous amount of RAM installed in the LFTB is an advantage, since one option is to use the RAM as intermediate storage for the captured data. The network adapters need to support the promiscous mode for capturing every network packet passing through the network (see also [Tan96]). They should match the surrounding network interface devices (e.g. network speed, duplex operation mode) to prevent a deterioration of the network performance. Data is captured on the data link layer of the ISO/OSI model of network traffic (see [Zim80]). The basis for the Live CD is the Debian Linux Distribution[3] which was hardened with only necessary software present (see [Tur05]). We add the software packages *bridge-utils*, *tshark* and *wireshark-common*, which provides the network packet sniffer *dumpcap*. The first package allows a detailed configuration of two network adapters to form a network bridge. In order to be as stealthy as possible, the spanning tree protocol is disabled. The software $dumpcap$ can act as a network packet capturing tool. Potential alternatives such as *ettercap* were not selected, because they offer the option of actively manipulating network packages and staging man-in-the-middle attacks. To achieve confidentiality and privacy, we use the software *TrueCrypt*[4], which provides a secure virtual drive. A self-developed shell script combines all the software components and provides the user with a screen dialogue based interface.
A useful addition when using the LFTB is the network tap (see also [Lai00]), which separates the upstream and downstream directions of data within the network. Most of the taps offer a read-only access, which is beneficial to forensic investigations, ensuring the stealth functionality and preventing manipulations by the capturing process during the data gathering phase DG (see 2.1). Also placing the network taps as part of the strategic preparation SP allows for a more economical use of the LFTB devices (see Section 5). This enables the investigator to deploy the LFTB without disconnecting the network connection on demand. It also prevents negative impacts to the availability of the network in case of hardware or software failure of the LFTB.
After inserting the LFTB media our boot shell script establishes the bridge (”br0”), using the two network adapters. The script then logs in the user ”lftb” and invokes the execution of the application script that forms the functionality of LFTB. The user is asked to verify the system time to assure a valid time base. The usage of the NTP network time is not an option as it would give away the presence of the LFTB. Although not currently implemented, in principle the employment of radio controlled clocks or GPS modules is possible.
After ensuring the time base, the script asks for the name of the investigator for use in transcripts and reports. In the next step the investigator has to chose a target for storing the captured packets. A menu allows for mass storage devices initialised for use with the LFTB. The script checks and warns the investigator, if data from previous sessions has been detected. With the highly favourable option of using a virtual secure drive being cho-

[2] http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html
[3] http://www.debian.org
[4] http://www.truecrypt.org/downloads

sen, this drive can also be created during this step. After that the investigator currently has five options for recording network data traffic:

- Capture complete traffic.
- Capture traffic to or from a particular MAC.
- Capture traffic using a custom filter.
- Capture traffic using a timer.
- Capture traffic with delayed start and timer.

These options are chosen to reflect different needs when being deployed at different locations (see Section 5). Especially the employment of custom filters (e.g. only recording traffic using the http protocol) allows for scalability with respect to respect to storage space limitations and for application in support cases. The options differ further in the network source and the start and length of the capturing process. All options, however, gather network data starting from the data link layer up to the application layer of the ISO/OSI Model (see also [Zim80]). Further options could easily be added, should the need for extension arise. Every network data recording is then secured in its integrity using the sha256 hashing algorithm. However this can not cover errors during the writing process to the HDD, because the hash is computed afterwards. In future releases the network packet capturing tool should be extended or replaced by a tool which supports internal hashing. The transcript contains timestamps for the start and the termination of the recording (see Figure 4). Those dates are based upon the time the investigator entered when starting the LFTB and are therefore independent of potentially maladjusted system clocks in the network, therefore providing proof within the limits of the correct clock setting of the investigator.

```
Linux forensic transparent bridge evidence storage
Starting time: Sun Mar 28 20:18:51 CEST 2010
Investigator: Hildebrandt
--------------------------------------------------------------------------------
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

Log item SHA256 hash: 717322afeb1490eaede00b4dfc40215046e60d3d9e3ab75cfe73fdbb1d
053e91
HMAC: 3d161b1c8168a313c8f2cd3a6c5e2b3b58b7507f12e8a091cad81183178b9837
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

--------------------------------------------------------------------------------
starting time: Sun Mar 28 20:18:55 CEST 2010
action: dumpcap -i br0 -w /root/data/1269800335.cap -a filesize:127004
exit time:Sun Mar 28 20:19:29 CEST 2010
result: /root/data/1269800335.cap
SHA256 hash: eb5b59ecf71ce7adbf9e8c08e1b75c904b9a0387ac73ebc8c324f375a6144295 /r
oot/data/1269800335.cap
dumpcap output:
Packets: 95648 Packets dropped: 0
--------------------------------------------------------------------------------
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

Log item SHA256 hash: a466dc5cdded8b9c5e1e59e20e54fff7379f6a398e7f652c6f80cbfa4d
92ab26
HMAC: 32066f6d65a8aa4b9b708c2cf13635d48220e8f5294ea1590804bf2eaa3afbd8
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

--------------------------------------------------------------------------------
MAC-table of br0 at Sun Mar 28 20:19:33 CEST 2010
port no mac addr is local? ageing timer 1 00:0a:8a:a2:30:b5 no 38.69 1 00:0c:29:
3a:86:af yes 0.00 2 00:0c:29:3a:86:b9 yes 0.00 1 00:12:43:30:c1:e7 no 43.66 1 00
:19:66:c3:47:1d no 73.59 1 00:1a:4f:85:0f:6d no 36.79 1 00:30:1b:b8:1e:6c no 8.6
2 1 00:80:c8:d7:ef:c5 no 35.41 1 40:00:04:11:6f:44 no 10.38 1 40:00:04:11:6f:46
no 10.18 1 40:00:04:11:6f:52 no 10.58 1 40:00:04:11:6f:7d no 10.78 1 40:00:04:11
```

Figure 4: Output of the protocol function of the LFTB

In the bottom of the figure the MAC table of the bridging mechanism is shown, which can be crucial when IP addresses need to be correlated to hardware MAC addresses. Confirming with forensic best practices every action the investigator performs is recorded together with command line parameters and the results. A sha256 hash sum and a sha256 hmac hash is being calculated for every item, thus ensuring $integrity$ and $authenticity$ of the collected data. In contrast to the hash of the network dumps, the hashes for each log item are computed internally before the storage process. Thus write errors can be detected.

5 Positioning of the Linux Forensic Transparent Bridge (LFTB)

The data collected by the LFTB depends heavily on the location within the network it is placed in. This placement is a vital part of the strategic preparation SP introduced in Section 2.1. Figure 5 shows a typical local network and potential locations of the LFTB.

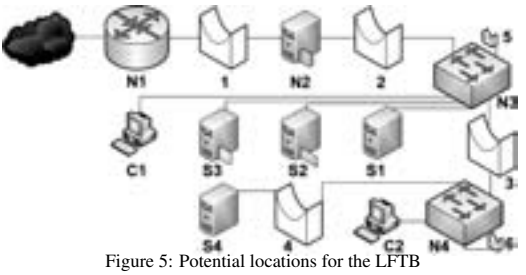

Figure 5: Potential locations for the LFTB

In this setup $N1$ represents a router connected to the Internet. $N2$ is a Firewall/VPN server. $N3$ and $N4$ represent network switches. $S1$ to $S4$ incorporate various dedicated servers. In an exemplary setup, $S2$ has a mail server functionality, with $S3$ acting as a file server. $S4$ is a general purpose server with high protection requirements. In this setup, $C1$ represents a client with normal and $C2$ a client with high protection requirements.
If the LFTB is placed at location 1, it records unfiltered network data traffic to and from the Internet. However, VPN connections are only recorded in their encrypted representation. If the LFTB is stationed at location 2, it captures filtered network data traffic to and from the internet. VPN data is captured in its unencrypted representation. If the functionality of the firewall $N2$ has to be investigated, two LFTBs on location 1 and location 2 are needed. If a LFTB is situated at location 3, the transition of network data between systems with normal protection requirements to systems with high protection requirements can be captured. To exclusively capture network data originating from or destined to server $S4$ with high protection requirements, a LFTB should be placed at location 4. Network taps (see Section 4) at the locations 1 to 4 should to be placed as part of the strategic preparation SP. If the switches $N3$ or $N4$ have a monitoring port, the LFTB could be connected to it.

6 Examples for using a Linux Forensic Transparent Bridge (LFTB)

In this section we show, using two scenarios, how a deployment of LFTBs collecting forensic sound evidence (i.e. adhering to the security aspects of $integrity$, $authenticity$) as well as conforming with $confidentiality$ and $privacy$ can aid in investigating both support cases (malfunctioning hard-/software components and operating errors) and intended malicious activities. The result in both cases is a capture file together with a sha256 hash sum ensuring $integrity$ as well as a report file with a hmac sum for every protocol entry ensuring $authenticity$ and a sha256 sum ensuring the $integrity$ of the report file. The output is placed in a secure virtual drive for $confidentiality$ and $privacy$, allowing access only to the investigator supplying the chosen password. Also we discuss the scalability of the LFTB for the chosen examples as a first evaluation of the concept of the LFTB.

Support case For the first scenario we assume that two systems within a network are accidentally assigned the same IP address. This seemingly trivial problem can prove to be a major problem especially in large intranetworks. In the network (see Figure 5) the task of distribution of IP addresses is assigned to the dhcp server $N2$. It uses static dhcp (i.e. each hardware MAC address is tied to a preconfigured IP address). A client using the Microsoft Windows XP operating system shows multiple alerts of IP address conflicts. A LFTB was

placed at location 3, from which the whole of the network traffic can be captured with all local IP addresses being present. The network traffic (DT_2, see Section 2.3) is recorded as part of the data gathering phase (see Section 2.1) using the LFTB.
Depending on the overall network bandwidth the scalability of the LFTB can be a factor. Since at location 3 the full bandwidth can be required, off-the-shelf PC hardware may not be sufficient to capture the full data content in time, especially if a secure virtual drive is used.
For the following investigation we select the tool "tshark[5]" from the class DPE (see Section 2.2) because it allows the dissection of the captured data in the following phase of data investigation DI to extract data confirming to the ISO/OSI transport layer (see [Zim80]). This extracts the MAC address data (DT_5, see Section 2.3) from the captured network data. The same tool can also be used to extract ISO/OSI gateway layer data (DT_5). In the phase of data analysis DA those two investigation results can be correlated. For this we also need the ARP tables[6] (DT_5) as collected by the LFTB. It in this case an IP address is used by two different MAC addresses. By using this information to correlate it with the data from the dhcp server, it shows which computer was using the IP wrongly. By having the MAC table from the LFTB it can be seen, in which network segment the computer that is using the IP address wrongly is located. This enables the investigator to locate the misconfigured system even without managed switches, witch also maintain a user accessible MAC table. Following the data analysis part, a detailed technical report in the documentation phase DO (see Section 2.1) which also includes all of the accompanying documentation has to be created.

Malicious activity In the second scenario, an unusual network activity has been observed. An LTFB was placed on location 2 (see Figure 5) during the strategic preparation SP (see Section 2.1). This location is chosen because the whole and unfiltered network data traffic (DT_2, see Section 2.3) is needed and no filtering decisions can be made as part of the operational preparation OP. The network data traffic is captured by the LFTB to secure evidence as part of the data gathering phase DG. At the chosen location the scalability of the LFTB is not likely to be an issue since the volume of network traffic passing from the intranet to the Internet will not overstrain the LFTB. In the following phase of data investigation DI, the communication protocol data DT_5 of the captured network data is dissected using the tool $tshark$. The following Figure 6 shows a sample of the output of $tshark$ when used in the data investigation mode, depicting communication protocol data (DT_5, see 2.3) on the transport layer according to [Tan96]).

TCP Conversations
Filter:<No Filter>

		<-		->		Total	
		Frames	Bytes	Frames	Bytes	Frames	Bytes
192.168.3.200:8822	<-> 192.168.1.14:3722	164	27540	163	10997	327	38537
192.168.3.200:8823	<-> 192.168.1.14:3124	71	14930	71	4802	142	19732
192.168.3.200:50251	<-> 192.168.1.14:11457	51	6378	71	4978	122	11356
192.168.3.200:52786	<-> 192.168.1.14:447	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:1398	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:3985	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:1021	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:268	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:479	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:605	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:742	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:288	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:1541	1	54	1	58	2	112

Figure 6: Communication protocol data analysis using $tshark$

Consequently the IP traffic together with port information is visible. It shows that one computer (192.168.3.200) establishes a lot of connections to different TCP ports of another computer (192.168.1.14) only transmitting one single network packet (a typical indicator

[5] see http://www.wireshark.org
[6] see also http://www.ietf.org/rfc/rfc826.txt

of a port scan). Also network traffic using the TCP ports 11457, 3124 and 3722 is detected. Since those are of no regular use on that computer, an investigation of the payload is inevitable as part of the data analysis phase DA. The tool $wireshark$ can be used on the packet level of the captured data from the LFTB. The following Figure 7 shows the output of a packet level analysis to extract user data DT_8 (see 2.3).

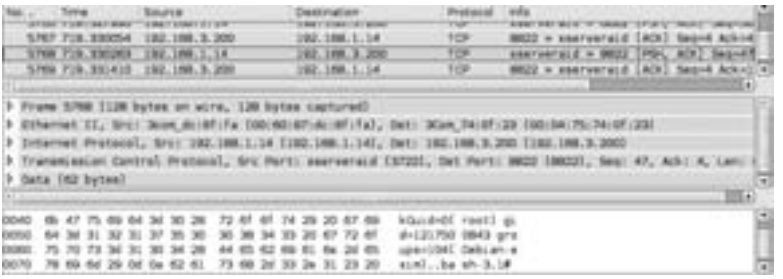

Figure 7: Network data packet level analysis using $wireshark$

The communication protocol data can be seen in the middle of the screenshot, sectioned into the different layers of the ISO/OSI model. In the *Ethernet II* entry the MAC address (DT_5) representing the layer 2 (i.e. the data link layer) is shown. The content of the network layer, the IP address, constituting DT_5 is shown on the entry *Internet Protocol*. The entry *Transport Control Protocol* shows the ports used and represents transport layer content (DT_5). The payload of the selected network data packet is shown on the lower part of the screenshot (DT_8). We select the forensic toolkit $PyFlag$[7], from the class DPE (see Section 2.2) because it enables the restoration of complete sessions (i.e. the combination of payload data).

Figure 8: Network session analysis using $pyflag$

The resulting output (see Figure 8) shows the execution of the script named $make\_me\_root$. Also, a malicious kernel module $enyelkm.ko$ is moved to the /etc directory on the target computer. In the data analysis DA phase it can be established, that shell commands are transmitted using those ports. As shown in Figure 7, this backdoor-ports were also utilised by the attacker. By having the complete network data recording, the progress of the attacker can be reconstructed, including which data was accessed by the intruder. A detailed report in the documentation phase DO (see 2.1) ends the investigation.

[7] http://www.pyflag.net

7 Testing the Linux Forensic Transparent Bridge (LFTB)

Testing of the LFTB covers the impact on the network by deploying one or more LFTB-devices and the quality of the collected evidence. The test concept includes throughput and latency, as well as the expected packet loss depending on the storage destination and the integrity and authenticity of the log-file.

LFTBs impact to the network For the impact the network transfer rate and the latency of packets were tested. Each test was performed for three setups: no LFTB deployed, LFTB connected to the monitor-port of the switch and LFTB in the bridging-mode.

performed test	reference without LFTB	LFTB at monitor port	LFTB in bridging mode
average ping latency	0.258ms	0.252ms	0.414ms
transferring time	22.655s	23.196s	22.918s

Table 1: Average ping latency and transferring time

Table 1 shows the average ping latency in the tests and the time to transfer a 256MB file via netcat through the network. The ping-latency has not changed in the first two cases and got bigger in bridging-mode. This is an expected result, because every OSI-Layer-2 device adds to the total latency. The network transfer rate has not changed significantly.

Quality of the collected evidence When capturing network traffic a certain amount of dropped packets, resulting from insufficient CPU-power or from the bandwith limitation of the storage device. Especially the ad hoc encryption consumes a lot of CPU-time. Our testing system was equipped with two Pentium II CPUs, each running at 450MHz and 384MB of RAM. The network connection was set to 100MBit.

	HDD+TrueCrypt (256MB file)	HDD (256MB file)	RAM-drive (128MB file)
captured/dropped (percentage)	139403 / 150853 (51.97)	210929 / 83075 (28.26)	139170 / 9603 (6.45)

Table 2: Captured and dropped packets

Table 2 shows the percentage of dropped packets. Our conclusion is that the testing system is insufficient for capturing the full 100mbit traffic. In each case a warning message was displayed to warn about the packet loss. In order to be able to capture the full traffic without dropping packets the prior-incident testing is a necessity. However, this packet loss only affected the recording, not the network itself. Further testing showed that newer systems are capable of recording the full 100mbit traffic into a RAM-drive (see Table 3).

System	RAM-drive (256MB file)	HDD (256MB file)	HDD+TrueCrypt (256MB file)
Pentium-M 1.6GHz	0	0.41	6.47
Athlon64 X2 2*2GHz	0	0.37	3.69
PhenomII X4 4*3.2GHz	0	0.01	0.13

Table 3: dropped packet percentage on newer systems (worst results)

The systems in Table 3 used different hard disk drives. For the PhenomII X4 an internal 3.5 inch SATA disk was used. The two other systems utilised a 2.5 inch SATA HDD, which was connected externally via USB. While capturing to the internal drive resulted in no or small amount of dropped packets, the USB device caused constant drop rates around 0.4 percent. However the realtime AES encryption needs sufficient processing power. The five year old hardware of the two slower systems is obviously not sufficient. The PhenomII is still too slow resulting in drop rates between 0.046 and 0.13 percent. The utilisation of special network adapters, which reduce the CPU load, might improve those results.

Reliability of the LFTB log file Although each item is secured by a sha256 hash it is very easy to replace an item with another one. Hence, an additional HMAC is used to provide authenticity. To prove the authenticity of the log item the output of dmidecode of the LFTB and the HMAC password is required.

8 Conclusion

In this paper we have proposed the Linux Transparent Forensic Bride (LFTB) as a network data gathering tool for the investigation of support cases and malicious activities. The LFTB allows for comprehensive and stealthy capturing of network data, ensuring integrity and authenticity of the data and, optionally but highly recommended, confidentiality and privacy. For its construction a self-developed holistic model of the forensic process was used, also firmly integrating accompanying documentation. We discussed the importance of the strategic preparation as an important part of forensic investigations to broaden the view of forensic investigations as a well defined way of data analysis not only limited to criminal proceedings. Especially when using the proposed LFTB the strategic preparation becomes paramount as the location of such a capturing device needs to be determined ahead of a potential incident.
The location management, the scalability to record potential large amount of data in quickly as well as issues with the key management for the hmac and the secure virtual drive containing the evidence constitute further work and testing areas.

Acknowledgement

The work in this paper for IT-forensics has been supported in part by the the Federal Office for Information Security (BSI). The authors wish to thank Mr. Carsten Schulz from the BSI for the initial inspiration and continued support.
The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

[Bis06] Matt Bishop. *Computer Security - Art and Science*. Number ISBN 0-201-44099-7. Addison-Wesley, 2006.

[BW06] Steve Bunting and William Wei. *The official EnCE Study Guide*. Number ISBN 978-0-7821-4435-2. Wiley Publishing Inc., 2006.

[Fre07] Felix Freiling. A Common Process Model for Incident Response and Digital Forensics. *Proceedings of the IMF2007*, 2007.

[Lai00] Brian Laing. Intrusion Detection Systems. Technical report, IBM Internet Security Systems, 2000.

[Nar10] Narus. Narus - Leader in Real-Time Traffic Intelligence. http://www.narus.com/, April 2010.

[New07] Robert C. Newman. *Computer Forensics - Evidence collection and Management*. Number ISBN 978-0-8493-0561-0. Auerbach Publications, 2007.

[Som06] Ian Sommerville. *Software Engineering*. Number ISBN 0-321-31379-8. Addison Wesley, 2006.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Number ISBN 0-13-394248-1. Prentice Hall International Inc., 1996.

[Tur05] James Turnbull. *Hardening Linux*. Number ISBN 978-1-59059-444-5. Apress, 2005.

[Zim80] Hubert Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transaction on Communications*, 1980.