A Labeled Data Set For Flow-based Intrusion Detection

Anna Sperotto, Ramin Sadre, Frank van Vliet, Aiko Pras

University of Twente
Centre for Telematics and Information Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
P.O. Box 217, 7500 AE Enschede, The Netherlands
{a.sperotto, r.sadre, a.pras}@utwente.nl,
d.f.vanvliet@student.utwente.nl

Abstract. Flow-based intrusion detection has recently become a promising security mechanism in high speed networks (1-10 Gbps). Despite the richness in contributions in this field, benchmarking of flow-based IDS is still an open issue. In this paper, we propose the first publicly available, labeled data set for flow-based intrusion detection. The data set aims to be *realistic*, i.e., representative of real traffic and *complete* from a labeling perspective. Our goal is to provide such enriched data set for tuning, training and evaluating ID systems. Our setup is based on a honeypot running widely deployed services and directly connected to the Internet, ensuring attack-exposure. The final data set consists of 14.2M flows and more than 98% of them has been labeled.

1 Introduction

Considering the increasing number of security incidents and discovered vulnerabilities per year [1], it is not surprising that intrusion detection (ID) has become an important research area in the last decade. A large number of ID techniques have been proposed and many of them have been implemented as prototypes or in commercial products. Moreover, the research community has recently focused on flow-based approaches.

When proposing a new intrusion detection system (IDS), researchers usually evaluate it by testing it on labeled (or annotated) traffic traces, i.e., traffic traces with known and marked anomalies and incidents [2]. Labeled traces are important to compare the performance of diverse detection methods, to measure parameter effectiveness and to fine-tune the systems. Ideally, a labeled traffic trace should have the following properties: it should be realistic (opposed to "artificial"), completely labeled, containing the attack types of interest and, not less importantly, publicly available. Despite the importance of labeled traces, research on IDS generally suffers of a lack of shared data sets for benchmarking and evaluation. Moreover, we have no knowledge of any publicly available *flow-based* traffic trace that satisfies all these criteria.

Several difficulties prevent the research community to create and publish such traces, in first place the problem of balancing between privacy and realism. It is natural that the most realistic traces are those collected "in the wild", for example at Internet service providers or in corporate networks. Unfortunately, these traces would reveal privacy sensitive information about the involved entities and hence are rarely published. On the other hand, artificial traces, i.e., traces that have not been collected but artificially generated, can avoid the problem of privacy but they usually require higher effort and deeper domain knowledge to achieve a realistic result. Moreover, labeling is a time consuming process: it could easily be achieved on short traces, but these traces could present only a limited amount of security events. Therefore, most publications use non-public traffic traces for evaluation purposes. The only notable exception is the well-known DARPA traces [3–5], which still are, despite their age, the only publicly available labeled data sets specifically created for intrusion detection systems evaluation.

In this paper, we present the first labeled flow-based data set. We describe how a suitable traffic trace can be collected and how it can be labeled. A flow is defined as "a set of IP packets passing an observation point in the network during a certain time interval and having a set of common properties" [6]. Our final data set will contain only flows and full-packet content will be used only as additional information source during the labeling process. Concentrating on flow-based data sets is in our opinion important for mainly two reasons. First, it substantially reduces the privacy concerns compared to a packet-based data set: since there is no payload, it is possible to deal with privacy issues by anonymizing the IP addresses. Second, several promising results have been recently proposed in the flow-based intrusion detection research community (for example, [7–9]): we feel that especially now there is the need for a shared flow data set.

Publications on the labeling of non-artificial data traces are rare and, as said, we are not aware of any concerned flow-based data sets. WebClass [10], a web-based tool that allows people to contribute to manually label traffic time-series, has not found much interest in the community. More effort has been done in the creation of artificial data sets. The already mentioned DARPA trace consists of generated traffic, equivalent to the traffic at a government site containing several hundreds of users on thousands of network hosts. In [11], the authors propose the synthetic generation of flow-level traffic traces, however without discussing its technical implementation. The malicious-traffic generation in the MACE [12] and FLAME [13] frameworks are performed by mixing background traffic with the output of different attack and anomalies modules. In MACE, it is up to the user to create a realistic traffic trace by providing an appropriate configuration file.

The remainder of the paper is structured as follows. In Section 2, we describe and discuss the data collection setup. The processing of the collected data and the labeling of the found intrusions and anomalies is described in Section 3. The resulting data set and its characteristics are discussed in Section 4. Finally, the paper concludes with Section 5.

2 Data collection

A proper measurement setup depends on the requirements we expect the labeled data set to meet. In our case, we want the data set to be *realistic*, as *complete* in labeling as possible, *correct*, achievable in a acceptable *labeling time* and of a sufficient *trace size*. In our research, we studied diverse approaches to data collection. In Section 2.1

we present our operational experience, explaining strengths and drawbacks of possible measurement infrastructures according to our data set requirements. In addition, Section 2.2 describes the technical details of our measurement setup.

2.1 Operational experience in data collection

This subsection will discuss the impact of both *measurement scales* and *flow collection location* on our requirements.

Measurement scale Flows can be collected at different measurement scales. The University of Twente has a 10 Gbps optical Internet connection with an average load of 650 Mbps and peaks up to 1.0 Gbps. Several hundred million flows are exported per day [8]. Traces collected on this network would for sure be realistic, but we cannot accomplish the goals of completeness in labeling and reasonable creation time. Labeling network-wide traces suffers of scalability issues.

Let us now concentrate on a small subnetwork that is primarily used by us for research purposes. Due to the limited number of users, it should be easy to distinguish trusted IP addresses from unknown ones, leaving out only a small fraction of suspicious traffic to be further analyzed. However, our findings show that more than 60% of the connections cannot easily be categorized as malicious or benign. Collecting on a small subnetwork ensures us to have a realistic data set, but, as for the network scale, it would be neither complete nor achievable in a short time.

A different setup, and the one that we finally chose, is based on monitoring a single host with *enhanced logging* specifically tuned to track malicious activities, e.g., a *honeypot*. A honeypot can be defined as an "environment where vulnerabilities have been deliberately introduced to observe attacks and intrusions" [14]. In this case, the trace size would be smaller and the labeling time limited. Moreover, an everyday service setup would ensure the traffic to be realistic. Finally, the logs would ensure us to achieve both completeness and correctness in labeling.

Flow collection location We have different options about the flow collection location, while we collect logs on our honeypot. A possibility is to collect the flows generated by a Netflow enabled router, in our case the University one. However, decoupling the flow creation from the log location introduces errors in measurements, such as timing delays and split TCP sessions. These issues make impossible to properly associate a security event with the flow that caused it. A data set based on these premises would have a serious lack in completeness and correctness. Another possibility is therefore to dump only the traffic reaching the honeypot and to create the flows off-line after the data collection is completed. This decision allows to have complete control over the flow-creation process, overcoming the problem of the session splitting.

Discussion We will now summarize the methods we studied for the data set collection, showing if they meet our requirements. Please note that, since monitoring the university network or a subnetwork does not scale, we did not explore further the options of online/offline flow creation. All our approaches ensure the trace to be realistic. Moreover, monitoring the University network or a subnetwork would also provide sufficiently large traces. After we measured the traffic reaching our honeypot, we can say that also in this case this requirement can be met. Regarding completeness and correctness, large network traces reduce the trust we can have on the labeled data set. The honeypot approach is more reliable since it offers additional logging information, but in this case, the flow collection/creation is crucial. As previously in this section, relying on external flow sources can introduce measurement errors. Creating the flows offline, on the other hand, allows to have flows that better suit the needs of a labeled data set. Finally, large infrastructures suffer of scalability in labeling time, while the honeypot setup overcomes this problem.

From this section, we can conclude that monitoring a single host that has enhanced logging capabilities is a promising setup for flow-based data sets.

2.2 Experimental setup

Our honeypot was installed on a virtual machine running on Citrix XenServer 5 [15]. The decision to run the honeypot as a virtualized host is due to the flexibility to install, configure and recover the virtual machine in case it is compromised. In addition, a compromised virtual machine can be saved for further analysis. Diverse scenarios are possible, in terms of number of virtual machines, operative systems and software. Our experimental setup consisted of a single virtual machine, on which we installed a Linux distribution (Debian Etch 4.0). In order to keep the setup simple, controllable and realistic, we decided not to rely on honeypot software, but to configure the host by ourselves. In particular, the following services have been installed:

– `ssh`: OpenSSH service [16]. Beside the traditional service logs, the OpenSSH service running on Debian has been patched in order to log sessions: for each login, the *transcript* (user typed commands) and the timing of the session have been recorded. This patch is particularly important to track active hacking activities.
– Apache web server: a simple webpage with a log in form has been deployed. We relied on the service logging capabilities for checking the content of the incoming `http` connections.
– `ftp`: The chosen service was `proftp` [17]. As for `http`, we relied on the `ftp` logs for monitoring attempted and successful connections. `proftp` uses the `auth/ident` service running on port 113 for additional authentication information about incoming connections.

Along with the service logs, we decided to dump all the traffic that reached the honeypot during our observation period. The virtual machine ran for 6 days, from Tuesday 23 September 2008 12:40:00 GMT to Monday 29 September 2008 22:40:00 GMT. The honeypot was hosted in the university network and directly connected to the Internet. The monitoring window is comprehensive of both working days and weekend days. The data collection resulted in a 24 GB dump file containing 155.2 million packets.

3 Data processing and labeling

The labeling process enriches the data trace with information about (i) the type and structure of the malicious traffic, (ii) dependencies between single isolated malicious activities. The latter is particularly important for a flow-based data set where by design no further detail on the content of the communication is available to the end user. In this

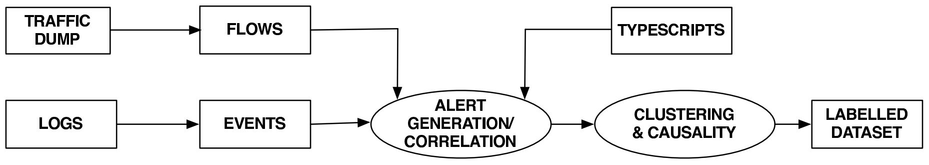


Fig. 1. From raw data (packets and logs) to the labeled data-set

section, we describe the processing of the collected traffic data and how the labeling information is generated and structured.

Figure 1 gives an overview on the whole data processing and labeling. As a first step, shown in the left part of the figure, the collected traffic dumps are converted into flows. In addition, the data extracted from the diverse log files is converted into a common format (log events) that simplifies the following processing steps. The resulting flow records and log events feed the alert generation/correlation process. Finally, a post-processing step generates additional information, namely the so-called cluster alerts and the causality information. The different steps are explained in the following Sections 3.1 through 3.4.

3.1 From packets to flows

The first step is the creation of flows from the traffic trace. In our data set, a flow closely follows the Netflow v5 definition and has the following form:

$$F = (I_{src}, I_{dst}, P_{src}, P_{dst}, Pckts, Octs, T_{start}, T_{end}, Flags, Prot)$$

where the unidirectional communication is defined by the source and destination IP addresses I_{src} and I_{dst}, the employed ports P_{src} and P_{dst} (in case of UDP/TCP traffic), and the level 3 protocol type $Prot$. The fields $Pckts$ and $Octs$ give the total number of packets and octets, respectively, in the data exchange; the TCP header flags are stored as a binary `OR` of the flags in all the packets of the flow (field $Flags$); the start and end time of the flow are given in T_{start}, respectively, T_{end} in millisecond resolution. The flow creation has been performed using a modified version of `softflowd`[18].

3.2 From log files to log events

Information about attacks against our honeypot can be extracted from the log files of the services we were monitoring. In order to simplify the alert generation process, the relevant data found in the various log files is converted into log events. A log event consists of the following information:

$$L = (T, I_{src}, P_{src}, I_{dst}, P_{dst}, Descr, Auto, Succ, Corr)$$

where T gives the timestamp of the attack (as found in the logs), I_{src} and P_{src} give the IP address and used port (if available) of the attacker, and I_{dst} and P_{dst} give the attacked IP address (our honeypot) and port number. In addition, a deeper analysis of the log files reveals whether an attack was automated or manual and succeeded or failed (flags $Auto$ and $Succ$, respectively). The field $Descr$ allows us to enrich the data set with additional information retrieved from the log files. The field $Corr$ is always initialized to $false$ and later used by the alert generation process.

3.3 Alert generation and correlation

Goal of this step is to generate so-called alerts and to correlate each generated alert to one or more flows. An alert describes a security incident and is represented by the following tuple:

$$A = (T, Descr, Auto, Succ, Serv, Type)$$

The fields T, $Descr$, $Auto$, $Succ$ are defined as for the log events (see Section 3.2). The field $Serv$ gives the service addressed by the security incident, for example `ssh` or `http`. The $Type$ field describes the type of the incident. The alert generation process consists of three steps that are explained in the following.

Alerts from log events For attacks toward the honeypot, alerts can be directly generated from the log events. The fields T, $Descr$, $Auto$, $Succ$ of an alert record are set to the values of the corresponding fields of the log event. The $Serv$ field is set accordingly to the destination port field of the log event, for example $Serv =$ `ssh` if $P_{dst} = 22$. The $Type$ field is always set to the value `CONN`, indicating that the alert describes a malicious *conn*ection attempt.

In order to correlate an alert with a flow, we have to find the flow that corresponds to the log event from which the alert has been generated. This is not a trivial task since the timing information extracted from a log file may not be aligned with the flow's one. In addition, we would like to not only correlate the incoming (as seen from the honeypot) flow to the alert but also the response flow of the honeypot.

We use the flows as starting point for the alert generation and correlation. This avoids that a flow is correlated to more than one alert. The resulting procedure for a service s is shown in Algorithm 1. As a first step, a best matching response flow is selected for each flow of the considered service (lines 3-6). The matching is made based on the flow attributes. If there are more than one candidate flows, the closest in (future) time is chosen. It is possible, nevertheless, that such a flow does not exist, for example in the case in which the target was a closed destination port. Since the flow tuple source/destination address/port may appear multiple times in the data set, the parameter δ ensures that an incoming flow is not correlated with a response too late in time. Values of δ from 1 to 10 seconds are possible.

After searching for a response flow, the algorithm proceeds with retrieving the best matching log event (lines 7-10). The log event must match the flow characteristics. The timing constraint in this case forces the log event to be in the interval between the beginning and the end of the flow. Moreover, since log files do not provide us with millisecond precision and multiple alerts can be generated by the same host in the same second, we require the matching log event to not have been consumed before (line 9). If the matching event exists, the algorithm will create the alert and correlate it with the flow and, if possible, with the response flow (lines 11-16). Finally, the log event will be marked as consumed (line 15).

Algorithm 1 Correlation procedure

```
1:  procedure ProcessFlowsForService ($s$ : service)
2:  for all Incoming flows $F_1$ for the service $s$ do
3:      Retrieve matching response Flow $F_2$ such as
4:      $F_2.I_{src} = F_1.I_{dst} \wedge F_2.I_{dst} = F_1.I_{src} \wedge F_2.P_{src} = F_1.P_{dst} \wedge F_2.P_{dst} = F_1.P_{src} \wedge$
5:      $F_1.T_{start} \leq F_2.T_{start} \leq F_1.T_{start} + \delta$
6:      with smallest $F_2.T_{start} - F_1.T_{start}$ ;
7:      Retrieve a matching log event $L$ such as
8:      $L.I_{src} = F_1.I_{src} \wedge L.I_{dst} = F_1.I_{dst} \wedge L.P_{src} = F_1.P_{dst} \wedge L.P_{dst} = F_1.P_{src} \wedge$
9:      $F_1.T_{start} \leq L.T \leq F_1.T_{end} \wedge$ not $L.Corr$
10:     with smallest $L.T - F_1.T_{start}$ ;
11:     if $L$ exists then
12:         Create alert $A = (L.T, L.Descr, L.Auto, L.Succ, s, \texttt{CONN})$.
13:         Correlate $F_1$ to $A$ ;
14:         if $F_2$ exists then
15:             Correlate $F_2$ to $A$ ; $L.Corr \leftarrow \mathbf{true}$ ;
16:         end if
17:     end if
18: end for
```

Alerts for outgoing attacks Some of the incoming attacks against the `ssh` were successful. As a consequence, the attacker used the honeypot machine itself to launch `ssh` scans and dictionary attacks against other machines. In order to generate alerts for these outgoing attacks, we have analyzed the typescripts of the `ssh` sessions. This allowed us to reconstruct the times and the destinations of the different attacks launched from the honeypot. Similarly to the previous step, we have used this information to find the corresponding flows and to correlate them to alerts of type `CONN`.

Alerts for side effects Several attacks towards and from the honeypot have caused non-malicious network traffic that we consider as "side effects". Most notably, `ssh` and `ftp` connection attempts caused ICMP traffic and traffic directed to the `auth/ident` service (port 113) of the attacker, respectively, the honeypot. Furthermore, one attacker installed an IRC proxy on the honeypot. For these flows, we have created alerts of type `SIDE_EFFECT` with $Serv = \texttt{ICMP}$, respectively, $Serv = \texttt{auth}$ or $Serv = \texttt{IRC}$.

3.4 Generation of cluster alerts and causality information

The alerts described in the previous section represent single security incidents and are directly correlated with one or more flows. In addition to those basic alerts, we also generate so-called cluster alerts and extra-information describing the causal relationships between alerts.

Cluster alerts are used to label logical groups of alerts. For example, in the case of an `ssh` scan consisting of 100 connection attempts, we create basic alerts of type `CONN` for each connection, plus *one* cluster alert of type `SCAN` for the entire scan operation. More formally, a basic alert A belongs to a scan alert C, if (i) the alert A has the same source IP and type as the other alerts in the cluster, and (ii) the alert is not later than γ seconds after the latest alert in the cluster. In our case we set $\gamma = 5$ seconds.

As a final step, we manually add causality information. In the current setup, that means that we have created (i) links between the alert representing an attacker's successful log-in attempt into the honeypot via `ssh` and all alerts raised by the attacker during that `ssh` session, and (ii) links between the alerts of the ICMP and `auth/ident` flows and the alerts of the `ssh` and `ftp` flows that caused them.

Our data set has been implemented in a MySQL database. The structure of the database reflects the alert and flow structure and the relations between these categories.

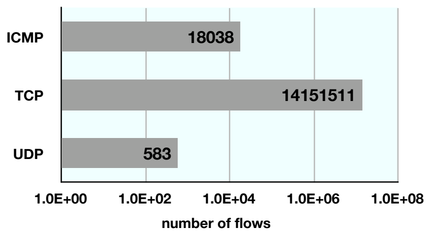
4 The Labeled Data Set

The processing of the dumped data and logs, collected over a period of 6 days, resulted in 14.2M flows and 7.6M alerts.

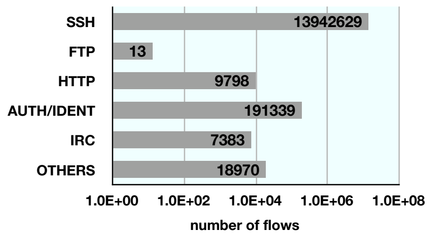
4.1 Flow and Alert breakdown

Figure 2(a) and 2(b) present a breakdown of the flows according to the level 3 protocols and the most active services, respectively.

At network level, the collected data set presents a subdivision in only three IP protocols: ICMP, TCP and UDP. The majority of the flows has protocol TCP (almost 99.9% of the entire traffic). A second slice of the data set consists of ICMP traffic (0.1%). Finally, only a negligible fraction is due to UDP traffic. The level 3 protocol breakdown is consistent with the services flow breakdown, shown in Figure 2(b). The honeypot most active service has been `ssh`, followed on the distance by `auth/ident`.



(a)



(b)

Fig. 2. Flow breakdown according to the level 3 protocol (a) and the service traffic (b)

Figures 3(a) and 3(b) show the alert breakdown according to malicious connections and scans on the most active services. `ssh` and `auth/ident` are the only services for which the honeypot has been both a target and a source. The honeypot received several thousand `ssh` connections and it has been responsible of millions of outgoing ones. On the contrary, it has been the target of extensive `auth/ident` requests, but it produced only 6 outgoing connection to port 113. As shown in Figure 3(b), the `ssh` alerts can be grouped into 45 scans, 10 incoming and 35 targeting remote destinations. We also have been the target of 4 `http` scans. None of the `ftp`, `auth/ident` or `irc` alerts can be clustered into scans.

4.2 Honeypot target and source behavior

Figure 4(a) presents the five most contacted ports on the honeypot. Port 113 (`auth/ident` service) is the most targeted one. Among the services we monitored, the most often contacted ones are `ssh` (port 22) and `http` (port 80). On these ports we actually received malicious traffic, confirming our initial idea that daily used services are normally target of attacks. The incoming traffic on port 68 is due to periodical renew of the dynamically assigned IP address (`dhcp`). Finally, we received traffic on port 137 due to the `netbios` protocol, to which we never responded.

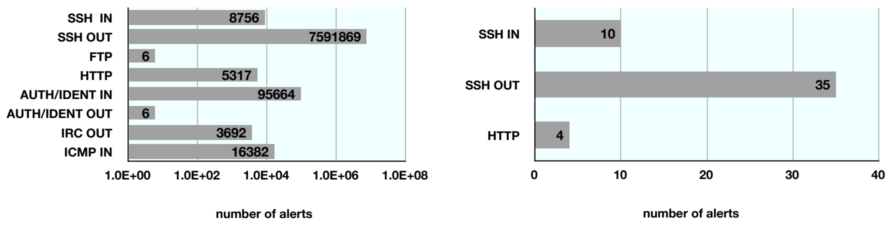
Figure 4(b) presents the top 5 ports contacted by the honeypot on remote destinations. The hit list is opened by `ssh`, confirming the alert breakdown in Figure 3. Traffic directed to port 67 has been generated by `dhcp` and it matches the incoming traffic on port 68. The traffic on port 53 (`dns`) was directed to the University `dns` servers. The outgoing traffic towards port 80 consists of only RST packets. The dumped traffic shows that remote hosts contacted us with SYN/ACK packets from port 80. Since the honeypot never initiated this interaction, it reset the connections. For its characteristics, this traffic seems not to be related to attacks: we suspected it is a form of background radiation [19, 20], or part of a SYN/ACK scan. Traffic to port 1 is negligible (`tcpmux`).

4.3 Discussion on the data set

This section will discuss the results we obtained as outcome of the correlation process, pointing out the characteristic of both the labeled and unlabeled traffic. Our correlation process succeeded in labeling more that 98.5% of the flows and almost the totality of the alerts (99.99%).

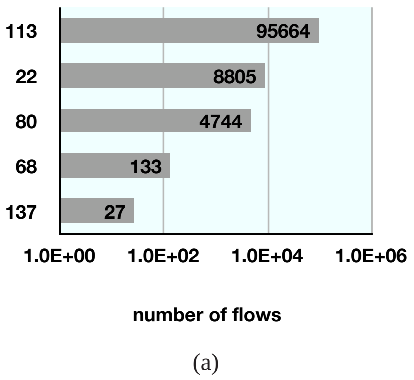
Malicious traffic All the labeled traffic is related to the monitored services. Since we did not interfere in any way with the data collection process, i.e., we avoided any form of attack injection and we did not advertise our infrastructure on hacker chats, we can assume that the attacks present in the data set reflect the situation on real networks.

The majority of the attacks targeted the `ssh` service and they can be divided into two categories: the automated and the manual ones. The first ones are well-known automated brute force scans, where a program enumerates usernames and passwords from large dictionary files. This attack is particularly easy to observe at flow level, since it



(a) (b)

Fig. 3. Alert repartition for basic (a) and cluster (b) types.



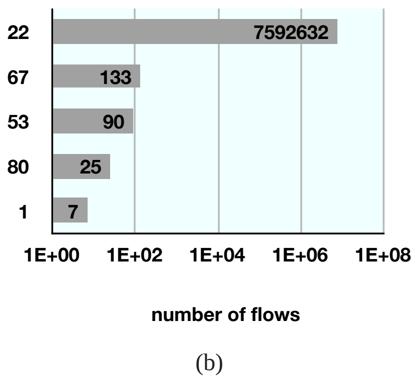


Fig. 4. Top 5 contacted ports. Figure (a) presents the incoming traffic, while Figure (b) the outgoing one (logarithmic scale).

generates a new flow for each connection. Attacks of the second type are manual connection attempts. There are 28 of these in our trace and 20 of them succeed.

The `http` alerts labeled in our data set are automated attacks that try to compromise the service by executing a scripted series of connections. These scans are executed using tools like Nikto and Whisker, both easily available online. Differently from the `ssh` connections, no manual `http` attacks are present.

Regarding the `ftp` traffic, the data set contains only 6 connections to this service on the honeypot, during which a `ftp` session has been opened and immediately closed. Even if the connections did not have a malicious content, this behavior could be part of a reconnaissance performed by an attacker gathering information about the system.

Side-effect traffic Part of the traffic in our data set is the side effect of attacks but cannot be considered by itself malicious. `auth/ident`, ICMP and `irc` traffic fall in this category. The scanning activities have been responsible of the majority of the flows to and from port 113. The service is supposed, indeed, to retrieve additional information about the source of a `ssh` connection. Regarding the ICMP traffic, more that 120,000 incoming packets have type *time exceeded* and *destination unreachable* and come from networks that the honeypot scanned.

The analysis of the honeypot showed that a hacker installed an IRC proxy that received chat messages from several channels. It does not appear anyway that any message has been sent from our honeypot or that the channels have been used for any malicious activity. Even if this traffic is due to an application that we did not originally install on our machine, we decided not to label it as malicious but as a side effect.

Unknown traffic and uncorrelated alerts For a small fraction of the data set, we cannot establish the malicious/benign nature of the traffic. This traffic consists of three main components: (i) `ssh` connections for which it has not be possible to find a matching alert, (ii) heterogeneous traffic, containing all the flows having as destination a closed honeypot port and (iii) some not malicious connections to the `http` and `vnc` services. The `vnc` service was accessible because of the XEN virtualization software running on the honeypot. Strangely, no attacker identified this service and tried to compromise it by using known vulnerabilities or performing a brute-force dictionary attack on the password . The `vnc` flows in the data set are originating from two hosts which contacted the service but did not complete the connection.

Regarding the alerts, for few of them (0.01%) no matching traffic has been found. This makes us aware that during the collection phase some packets that reached our honeypot have not been recorded by tcpdump.

5 Conclusions

The main contribution of this paper is to present the first labeled data set for flow-based intrusion detection. While approaching the problem of creating a labeled data set, we consider the choice of the proper data collection infrastructures a crucial point. It has indeed impact not only on the feasibility of the labeling, but also on the reliability of the result. We studied several data collection infrastructures, enlightening strengths and drawbacks. We conclude that, in the case of flow-based data sets, the most promising measurement setup is monitoring a single host with enhanced logging capabilities. In the specific context of this experiment, the host was a honeypot. The information collected permitted us to create a data base of flows and security events (alerts).

The paper also describes a semi-automated correlation process that matches flows with security events and, in addition, reflects the causality relations between the security events themselves. The results of the correlation process show that we have been able to label more that 98% of the flows in the trace. The correlation process also proves that, although we limited our measurements to a single host, labeling remains a complex task that requires human intervention.

The approach we propose in this paper allows us to capture unexpected security events, such as the behavior of a compromised host. However, the presented trace mainly consists of malicious traffic. For evaluation of an IDS, this means that our data set allows to detect false negatives but not false positives. In the future, we aim at extending our monitoring setup to more challenging scenarios. An example would be a server that is daily accessed by benign users. Monitoring such a server would result in a trace with a more balanced content of malicious and normal traffic. We believe that an approach similar to the one we presented in this paper will be useful also in this scenario. Finally, the data set is available on e-mail request to the authors.

Acknowledgments This research has been supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. CERT Coordination Center. http://www.cert.org/certcc.html (Jan. 2009)
2. Mell, P., Hu, V., Lippmann, R., Haines, J., Zissman, M.: An overview of issues in testing intrusion detection systems. Technical Report NIST IR 7007, National Insititute of Standards and Technology (June 2003)

3. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., S.E.Webster, Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: Proc. of the DARPA Information Survivability Conf. and Exposition (DISCEX '00). (2000)
4. Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks **34** (2000)
5. Haines, J., Lippmann, R., Fried, D., Zissman, M., Tran, E., Boswell, S.: 1999 DARPA Intrusion Detection Evaluation: Design and Procedures. Technical Report TR 1062, MIT Lincoln Laboratory (February 2001)
6. Quittek, J., Zseby, T., Claise, B., Zander, S.: Requirements for IP Flow Information Export (IPFIX). RFC 3917 (Informational)
7. A. Lakhina, M. Crovella, C.D.: Characterization of network-wide anomalies in traffic flows. In: Proc. of 4th ACM SIGCOMM Conf. on Internet measurement (IMC '04). (2004)
8. Sperotto, A., Sadre, R., Pras, A.: Anomaly characterization in flow-based traffic time series. In: Proc. of the 8th IEEE International Workshop on IP Operations and Management, IPOM 2008, Samos, Greece. Lecture Notes in Computer Science (September 2008)
9. Strayer, W., Lapsely, D., Walsh, R., Livadas, C. In: Botnet Detection Based on Network Behavior. Volume 36 of Advances in Information Security. (2008)
10. Ringberg, H., Soule, A., Rexford, J.: Webclass: adding rigor to manual labeling of traffic anomalies. SIGCOMM Computer Communication Review **38**(1) (2008)
11. Ringberg, H., Roughan, M., Rexford, J.: The need for simulation in evaluating anomaly detectors. SIGCOMM Computer Communication Review **38**(1) (2008)
12. Sommers, J., Yegneswaran, V., Barford, P.: A framework for malicious workload generation. In: Proc. of the 4th ACM SIGCOMM Conf. on Internet measurement (IMC '04). (2004)
13. Brauckhoff, D., Wagner, A., Mays, M.: Flame: a flow-level anomaly modeling engine. In: Proc. of the Conf. on Cyber security experimentation and test (CSET'08). (2008)
14. Pouget, F., Dacier, M.: Honeypot-based forensics. In: Asia Pacific Information technology Security Conference (AusCERT '04). (May 2004)
15. 5, C.X.: http://www.citrix.com/ (April 2009)
16. OpenSSH: http://www.openssh.com/
17. `proftp`: http://www.proftpd.org/
18. Softflowd: http://www.mindrot.org/projects/softflowd/ (April 2009)
19. Moore, D., Shannon, C., Brown, D., Voelker, G., Savage, S.: Inferring internet denial-of-service activity. ACM Trans. Comput. Syst. **24**(2) (2006)
20. Pang, R., V.Yegneswaran, Barford, P., V.Paxson, Peterson, L.: Characteristics of internet background radiation. In: Proc. of the 4th ACM SIGCOMM Conf. on Internet measurement (IMC '04). (2004)