Unveiling Flat Traffic on the Internet: An SSH Attack Case Study

Mattijs Jonker, Rick Hofstede, Anna Sperotto and Aiko Pras
Design and Analysis of Communication Systems (DACS)
Centre for Telematics and Information Technology (CTIT)
University of Twente, Enschede, The Netherlands
Email: {m.jonker, r.j.hofstede, a.sperotto, a.pras}@utwente.nl

***Abstract*—Many types of brute-force attacks are known to exhibit a characteristic ‘flat’ behavior at the network-level, meaning that connections belonging to an attack feature a similar number of packets and bytes, and duration. Flat traffic usually results from repeating similar application-layer actions, such as login attempts in a brute-force attack. For typical attacks, hundreds of attempts span over multiple connections, with each connection containing the same, small number of attempts. The characteristic flat behavior is used by many Intrusion Detection Systems (IDSes), both for identifying the presence of attacks and – once detected – for observing deviations, pointing out potential compromises, for example. However, flatness of network traffic may become indistinct when TCP retransmissions and control information come into play. These TCP phenomena affect not only intrusion detection, but also other forms of network traffic analysis. The contribution of this work is twofold. First, we analyze the impact of retransmissions and control information on network traffic based on traffic measurements. To do so, we have developed a flow exporter extension that was deployed in both a campus and a backbone network. Second, we show that intrusion detection results improve dramatically by up to 16 percentage points once IDSes are able to ‘flatten’ network traffic again, which we have validated by means of analyzing log files of almost 60 hosts over a period of one month.**

I. Introduction

Flow monitoring has become the prevalent approach for monitoring larger-scale, high-speed networks [1]. By aggregating packets into flows, which are defined as sets of packets that pass by an observation point in a network during a certain time interval [2], flow monitoring is typically said to be more scalable than monitoring approaches that rely on storing and analyzing individual packets. The fact that many high-end packet forwarding devices are already shipped with support for flow export protocols like NetFlow [3] and IPFIX [2], makes flow monitoring a cost-effective monitoring approach at strategic observation points in the network. Flow export can be applied to a variety of monitoring applications, such as network security, Quality of Service (QoS) monitoring, and accounting [4]. The scalability advantages of flow monitoring also come at a cost, however. For example, it is widely known that measurement artifacts may be present in flow data that affect the accuracy of the data in a negative way [5]. Also, by design, all packets that belong to a particular flow are accounted in the same way, making it impossible to differentiate classes of packets within the same flow (e.g., retransmitted packets).

The problem of having every packet in a flow accounted in the same way recently became apparent in our flow-based SSH Intrusion Detection System (IDS) SSHCure [6].[1] SSHCure is able to identify network scans, brute-force attacks using *dictionaries* (lists of frequently-used login credentials) and most important, compromises, which may be the result of a brute-force attack. Similar to other works in the area of SSH dictionary attack detection, SSHCure is built upon the assumption that network traffic of the *brute-force* phase of these attacks exhibits a characteristic ‘flat’ behavior. This means that multiple flows belonging to the *brute-force* phase of an attack are alike in terms of the featured number of packets and bytes, and duration [6]–[10]. The detection of the *compromise* phase is then performed by detecting deviations from the flat traffic of the *brute-force* phase. However, real-world deployments of SSHCure in various networks, ranging from smaller Web hosting companies to nation-wide backbones, have shown that this approach fails for many attacks from especially far-away countries. This is because network traffic of dictionary attacks is not always as flat as one would expect, an observation that is supported by the datasets used in [7]. Investigation of the network traffic at the packet-level has revealed that TCP retransmissions and other TCP control information are the suspected causes, which are typically not identifiable at the flow-level. Other approaches to flow-based *brute-force* and *compromise* detection exist, such as [11], but also suffer from TCP phenomena like retransmissions.

Only two works so far have studied behavior of TCP retransmissions and control information on the Internet. Both works, published in the 1990’s, focus on the behavior of TCP variants in terms of congestion control when facing packet loss, reordering or unexpected delays [12], [13]. The retransmissions in these works are introduced by means of simulating the aforementioned events, rather than measuring their occurrence on the Internet. To the best of our knowledge, there are no recent Internet measurements on the number of retransmissions and other TCP control packets. This work bridges this gap.

In this work, we study the impact of TCP retransmissions and other control information packets on network traffic in general and SSH intrusion detection in particular, both in a

[1]https://github.com/sshcure/sshcure/

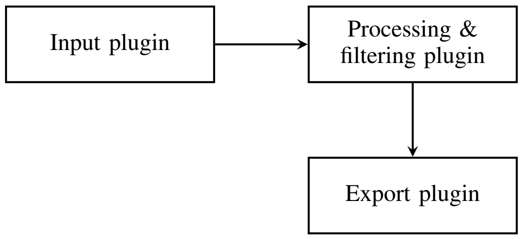

Fig. 1. INVEA-TECH FlowMon platform architecture, as of FlowMon Probe 6.x.

campus and a backbone network. To do so, we have extended the set of fields exported per flow, commonly referred to as IPFIX Information Elements (IEs), to include per-flow statistics on retransmissions and control information. Measurements have shown about 3 TiB of retransmissions per month on typical campus and backbone network links, accounting for roughly 1% of all TCP traffic over these links. This underlines the hypothesis that supposedly flat traffic may become non-flat, ergo indistinct. We study the effects of retransmissions and control information packets on SSH dictionary attacks, and compare the detection results of a state-of-the-art dictionary attack detection algorithm when the added information is used for detection instead of only traditional packet and byte counters in flow records. Our analysis shows that any flow-based IDS may profit from per-flow statistics on retransmissions and control information, as many previously undetected (non-flat) attacks can be revealed, resulting in an increase of 16 percentage points in terms of true detections when using a state-of-the-art SSH *brute-force* phase detection algorithm.

The organization of this paper is as follows. In Section II, we provide background information on TCP retransmissions and control information, and describe how TCP can affect the 'flatness' of network traffic. Then, in Section III, we discuss the defined IPFIX IEs and the implementation needed for measuring TCP retransmissions and control information. A description of the acquired datasets and measurement results are provided in Section IV. In Section V, we validate this work in the context of SSH intrusion detection, after which we draw our conclusions and state future work in Section VI.

II. Background

To facilitate reliable data delivery between endpoints, TCP uses a cumulative acknowledgement scheme in which sequence and acknowledgement numbers are used to signal the reception of data. In the absence of any feedback from the data receiver, a Retransmission TimeOut (RTO) is used to ensure delivery, which is based on the estimated Smoothed Round-Trip Time (SRTT). Due to unexpected delays or reordering of packets in the network, retransmissions can occur spuriously. For example, when a packet or its acknowledgement is delayed unexpectedly rather than lost, the RTO timer expires and the packet is retransmitted. Also, a fast retransmission may be sent when a certain number of consecutive duplicate acknowledgements is received, signalling the potential loss of packets to the sender. Due to reordering of packets, duplicate acknowledgements may be sent even though no packet has gotten lost. These duplicate acknowledgements can trigger a spurious fast retransmission. In both examples, spurious retransmissions and their duplicate acknowledgements cause additional packets and bytes in network traffic and hence affect the potential flatness of a connection.

To optimize network throughput while avoiding congestion or overloading an endpoint, TCP uses several techniques, such as flow control, based on a sliding window, and the *delayed ACK* mechanism. To realize flow control, the *receive window* needs to be signalled from receiver to sender, and under the *delayed ACK* mechanism, data acknowledgements are held back for a brief delay to save overhead. If data or additional control information becomes available during the delay, the held back acknowledgement can be combined with this information. For some forms of control information, such as data acknowledgements and *receive window* changes, the *delayed ACK* mechanism and circumstances dictate whether a dedicated packet is sent to carry the control information to the endpoint. For example, if during a *delayed ACK* data is pushed down from the application-layer, the held back acknowledgement can be *piggybacked* with a data packet. This prevents sending a dedicated acknowledgement with no payload. Also, the *delayed ACK* mechanism allows for the cumulative acknowledgement of two data packets received in rapid success. This too saves sending a dedicated packet. If the *receive window* changes at the receiver, this information can be combined with a held back acknowledgement, again saving a dedicated packet. Sometimes, however, the *receive window* expands when there is no data to acknowledge, in which case a dedicated *window update* needs to be sent. Whether or not such dedicated packets are sent affect the flatness of network traffic.

There are also types of control information that are always sent in separate packets: *Zero Window probes* and responses, *KeepAlive Probes* and responses, and `RST` packets. Also, depending on the TCP implementation, a three-way `FIN` close sequence may not be supported, thereby potentially introducing an additional packet during the connection termination. Any of these additional packets obviously affect the flatness of network traffic as well.

It is important to note that the presence of the aforementioned situations mostly depends on network conditions, resource availability and scheduling on endpoints, whether or not there is data to send or acknowledge, and timing.

III. Implementation

To export information that allows for the discrimination of TCP retransmissions and control information in flow data, several IPFIX Information Elements (IEs) were defined and implemented as part of a flow Metering Process. This section describes these IEs and the accompanying implementation.

We have defined IPFIX IEs for each of the TCP protocol phenomena discussed (in italics) in Section II. To facilitate the export of these IEs, we have developed an extension to INVEA-TECH's FlowMon flow exporter. This platform was

TABLE I
DATASETS

Dataset	Period	Duration	Packets	Bytes	Flows	Retransmissions		Control Information	
						Packets	*Bytes*	*Packets*	*Bytes*
UT	July / August 2014	31 days	370.73 G	291.64 TiB	7.35 G	5.30 G (1.43%)	2.83 TiB (0.97%)	100.50 G (27.11%)	4.30 TiB (1.47%)
CESNET	August / September 2014	31 days	257.38 G	227.67 TiB	3.57 G	8.29 G (3.22%)	2.78 TiB (1.22%)	83.61 G (32.48%)	3.48 TiB (1.53%)

chosen because of its highly customizable plugin architecture, and because we have full control over it in our networks. The complete architecture is shown in Fig. 1. It is based on plugins for data input, flow record processing & filtering, and export. Input plugins process data from a given source, such as one or more line cards, and are responsible for creating flow cache entries, one entry per active flow. Process plugins allow for the manipulation of these cache entries once these entries have been created. The process plugin type is best suited for program logic that does not necessarily require a packet's payload anymore. The export plugin is responsible for exporting cache entries by sending flow records to a collector using NetFlow or IPFIX. From within these plugin types, actions can be hooked to events such as flow entries being added to, updated in or expired from the flow cache. Among these actions is the filtering of flow cache entries to prevent them from being exported.

Our extension comes in the form of an input plugin, because it evaluates the payload of packets. The plugin measures TCP retransmissions and control information packets, and stores and maintains related counters in the flow cache. To recognize these particular packets, TCP conversations are analyzed in real-time by evaluating sequence and acknowledgement numbers, timestamps, flags, receive window sizes, and payload sizes. This implementation is heavily based on the TCP packet dissector used by *Wireshark*.[2]

For the TCP analysis to be accurate, it is crucial that packets in both directions of a TCP conversation pass through the observation point. Otherwise, the housekeeping of sequence and acknowledgement numbers may be affected, which obviously impairs the analysis. The same is true when packets are lost downstream of the observation point. We are also aware of the fact that the TCP packet dissector used by *Wireshark* cannot but misclassify packets in its on-the-fly analysis in some cases, especially when packets are reordered. To optimize our plugins to work on high-speed links, e.g., of *10 Gbps* and higher, we accept these exceptional cases for the sake of performance.

IV. Measuring TCP Retransmissions & Control Information

Our first step towards understanding the impact of TCP retransmissions and control information is to measure it in two networks that are different in nature. Two datasets were collected, as shown in Table I, consisting of only TCP flow data. Dataset *UT* was collected on the campus network of the University of Twente (UT). This network features a publicly routable /16 network address block with connections to faculty buildings, student and staff residences, etc. Due to the residential aspect of the campus network it also routes private c.q. non-academic Internet traffic. Furthermore, the campus network houses mirror servers for popular open-source software, such as Ubuntu. Dataset *CESNET* was collected on a backbone link of the Czech National Research and Education Network (NREN), specifically the link between CESNET and the 'commercial Internet'. Due to the academic nature of these networks, the relative amount of traffic during summer holidays is considerably lower than during working days.

[2]http://www.wireshark.org/

TABLE II
DISTRIBUTION OF RETRANSMITTED PACKETS AND BYTES

Dataset	Retransmissions		Fast Retransmissions	
	Packets	*Bytes*	*Packets*	*Bytes*
UT	95.50%	89.54%	4.50%	10.46%
CESNET	97.87%	91.27%	2.13%	8.73%

The remainder of this section is organized in two parts. First, in Section IV-A, we analyze retransmissions and control information in detail based on our measurements. After that, we perform a similar analysis only for SSH traffic in Section IV-B, given that the validation of this work (Section V) will be performed in the context of SSH intrusion detection.

A. *Overall Traffic*

Details on the number of retransmitted packets and bytes, and the amount of control information in terms of packets and bytes are shown in Table I. Several observations can be made. On the one hand, TCP control information is mostly visible in terms of packets. On the other hand, retransmissions contribute more towards the percentage of bytes. Another observation is that there are many more packets with control information than there are retransmitted packets. This is mainly because many control information packet types, such as those that result from the *delayed ACK* mechanism, are sent under all network conditions, while retransmissions appear more frequently during network congestion, for example.

The distribution of retransmission types in terms of packets and bytes is shown in Table II. As can be observed, most retransmissions are regular retransmissions. Also, for each dataset, the fraction of the total number of bytes for the fast retransmission type is higher than the packet fraction.

TABLE III
DISTRIBUTION OF CONTROL INFORMATION PACKETS

Type	**Dataset**	
	UT	*CESNET*
Duplicate ACK	5.24%	1.77%
Non-piggybacked ACK	7.61%	11.71%
Consecutive empty ACK	83.13%	80.60%
Window Update	2.02%	1.88%
Zero Window Probe (ZWP)	$< 0.01\%$	0.01%
ZWP response	$< 0.01\%$	$< 0.01\%$
`RST`	0.87%	2.59%
Four-way close packet	0.10%	0.21%
KeepAlive Probe	0.54%	0.74%
KeepAlive Response	0.48%	0.48%

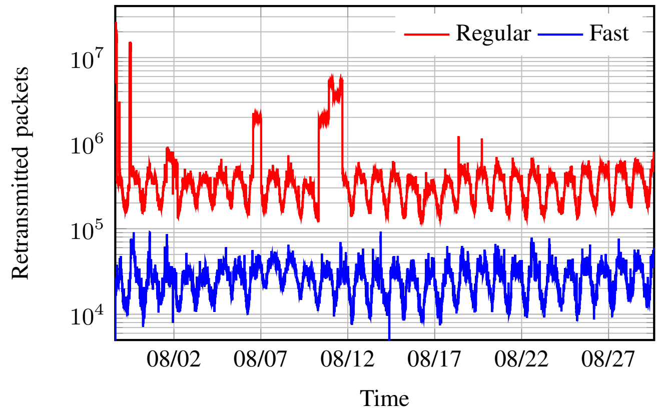

Fig. 2. Retransmissions over time.

We believe this is because regular retransmissions can also contain no payload, e.g., retransmissions of empty TCP `SYN` and `FIN` segments bring down the average number of bytes per retransmitted packet. Considering the *UT* dataset, it shows that while 4.50% of retransmitted packets are of the fast type, these do account for 10.46% of the number of retransmitted bytes. For the *CESNET* dataset, these numbers are 2.13% and 8.73%, respectively.

The number of retransmitted packets and fast retransmitted packets within every five-minute interval in the 31 days of the *UT* dataset is shown in Fig. 2. A diurnal pattern can be clearly identified, which follows the working hours at faculty buildings, and the presence of on-campus residents. While Table I provides absolute numbers, and as such is not specific about the points in time at which events occur, Fig. 2 shows that retransmissions occur at any time of the day. The two outlying groups of retransmitted packets around *5 Aug 18:00* and *10 Aug 18:00* coincide with severe SSH dictionary attacks from China that involve many retransmissions, which makes these anomalies visible in our measurements. These attacks will be discussed later, as part of the case study in Section V.

The distribution of the various types of control information packets is shown in Table III. As can be seen, packets related to the *delayed ACK* mechanism, i.e., *non-piggybacked ACKs*

TABLE IV
TCP RETRANSMISSIONS & CONTROL INFORMATION – SSH

Dataset	**Retransmissions**		**Control Information**	
	Packets	*Bytes*	*Packets*	*Bytes*
UT	1488.18 M (9.53%)	167.36 GiB (1.45%)	3269.24 M (20.93%)	145.19 GiB (1.26%)
CESNET	153.54 M (2.10%)	25.44 GiB (1.54%)	1767.31 M (24.15%)	76.78 GiB (4.64%)

TABLE V
DISTRIBUTION OF RETRANSMITTED PACKETS AND BYTES – SSH

Dataset	**Retransmissions**		**Fast Retransmissions**	
	Packets	*Bytes*	*Packets*	*Bytes*
UT	99.71%	96.47%	0.29%	3.53%
CESNET	99.87%	99.07%	0.13%	0.93%

and *consecutive empty ACKs*, account for large percentages of the total number of control information packets in each dataset. For example, *non-piggybacked ACKs* take up 7.61% and 11.71% in *UT* and *CESNET*, respectively. Another example is the *consecutive empty ACK*, with 83.13% in *UT* and 80.60% in *CESNET*.

Given the significant presence of TCP retransmissions and control information in our measurements in two networks that are different in nature, we conclude that these packets are omnipresent on the Internet. Also, we believe to have demonstrated that the flatness of originally flat network traffic on the Internet is likely affected by this omnipresence, as theorized in Section II.

B. SSH Traffic

The SSH traffic considered in this work was obtained by filtering the datasets presented in Table I for traffic on port 22, yielding 11.29 TiB of traffic for *UT* and 1.62 TiB for *CESNET*. Details on the number of retransmissions and control information packets and bytes are shown in Table IV. Several observations can be made when comparing the SSH traffic to the overall traffic. First, for *CESNET*, the relative percentage of retransmissions is lower in the SSH-only traffic than in the overall traffic, at 2.10% versus 3.22%. For *UT*, however, it is much higher, namely 9.53% versus 1.43%. This is because the *UT* dataset contains several large-scale SSH attacks, as discussed previously alongside Fig. 2. Second, control information in the SSH datasets is more dominant than retransmissions in terms of packets and bytes, which is similar in the overall traffic. Third, considering that the overall traffic in *UT* is only 50% larger than in *CESNET* in terms of bytes (from Table I), the relative amount of SSH traffic in *UT* is much larger than in *CESNET*.

As for retransmissions in SSH traffic, the distribution of these in terms of packets and bytes is shown in Table V. Compared to the distribution of retransmissions in the overall traffic, it can be observed that a higher percentage in the SSH traffic is of the regular retransmission type. In the

TABLE VI
DISTRIBUTION OF CONTROL INFORMATION PACKETS – SSH

Type	**Dataset**	
	UT	*CESNET*
Duplicate ACK	1.87%	2.70%
Non-piggybacked ACK	7.76%	63.63%
Consecutive empty ACK	89.58%	25.73%
Window Update	0.37%	0.35%
Zero Window probe	0.00%	$<$ 0.01%
ZWP response	0.00%	$<$ 0.01%
RST	0.31%	5.43%
Four-way close packet	0.09%	1.52%
KeepAlive probe	0.00%	0.35%
KeepAlive response	0.00%	0.31%

UT dataset, only 0.29% of retransmissions are classified as fast retransmissions, in contrast to a figure of 4.50% in the respective overall traffic. For *CESNET*, these numbers are 0.13% and 2.13%. Relative differences between the overall traffic and the respective SSH-only traffic thus follow the same trend. We believe that this is the case because it is less common for SSH connections to have four or more consecutive packets with payload sent by one endpoint within a short period. In other words, there are not enough consecutive data packets to trigger a fast retransmission.

The distribution of control information in SSH traffic is shown in Table VI. This distribution features several key differences compared to the full datasets (see Table III). A prime example is the significantly lower number of packets related to *KeepAlive*, especially for *UT* where there are none at all. A possible explanation for this is that the majority of SSH connections is short-lived, or otherwise active enough to not trigger the TCP *KeepAlive* timer, which is typically in the order of hours [14]. Another observation is that while the distribution of control information types is very similar within the full datasets collected on different networks, this is not the case anymore for the SSH datasets. For example, in the *UT* dataset, 0.30% of all SSH packets are *RSTs*, while *RSTs* account for 5.43% in *CESNET*. We believe that this is due to increased scanning activity. Also, for *CESNET*, *non-piggybacked ACKs* are at a staggering 63.63%, whereas in *UT* they account for only 7.76%. We believe these differences stem from the fact that a lot more data is sent within SSH connections on the UT network. Furthermore, for the *UT* and *CESNET* datasets it can be seen that *four-way close* features only very small percentages of control information packets, namely 0.09% and 1.52%, respectively. This leads us to believe that SSH network traffic is typically not affected much by this type of control information.

V. VALIDATION

In this section, we quantify and study the effects of TCP retransmissions and control information on flow analysis applications, in the context of a flow-based SSH intrusion detection case study. The case study used for validation is presented in Section V-A. The validation methodology is discussed in Section V-B. Finally, in Section V-C, we present the validation results.

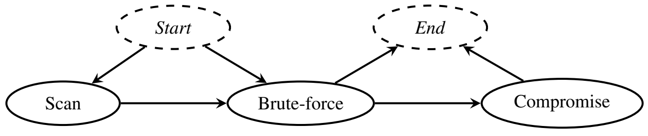

Fig. 3. Dictionary attack phases, from [1].

A. Case Study: SSH Intrusion Detection

The flow-based detection of dictionary attacks is typically performed by comparing the characteristics of two or more flow records to identify possible attacks. In [7], it is shown that these attacks typically consist of three phases, as shown in Fig. 3, which feature specific flow-level characteristics. During the *scan* phase, an attacker probes for the presence of specific services on one or more hosts in a network. During the *brute-force* phase, a high-intensity dictionary attack is performed on one or more targets on which the service is found active. The *brute-force* phase typically contains many flow records with an equal number of *Packets-Per-Flow* (PPF), since flows featuring an equivalent number of login attempts between the same client and server typically consist of the same number of packets. Should a compromise ensue, the *compromise* phase is reached.

In this work, we focus on the *brute-force* phase, since it is the only phase where 'flat' traffic should be predominant. The concept of an equal number of PPF, i.e., flat traffic, for brute-force attacks detection forms the basis of the state-of-the-art *brute-force* phase detection algorithm presented in [7], which considers the number of PPF in consecutive flow records. The algorithm starts with a preselection of source and destination IP address pairs for which flow records have a PPF value of $x \in [11, 51]$. For each of these preselected address pairs, the most frequently used PPF value is taken as the *baseline* for determining brute-force behavior. This *baseline* is then used for comparing consecutive flow records with identical PPF values to. If at least N consecutive flows feature the *baseline* number of PPF, a brute-force attack is recognized. We set the threshold N to five[3], and the result of the detection algorithm is a list of attacks. In the remainder of this work, we define an attack as a set of one or more targets featuring brute-force behavior for a given attacker, i.e., where every target in the set has reached N. A tuple is defined as a pair of attacker and target, such that every attack consists of one or more tuples.

If more than N flow records feature the same number of PPF, a dictionary attack is detected. On the one hand, false negatives, i.e., undetected attacks, can occur in this context when dictionary attack flows end up with diverse PPF values, causing the threshold N to not be reached, even though

[3]Note that five consecutive flow records with the same number of PPF would represent 15 failed login attempts in a benign situation, as explained in [7], which we consider highly unlikely.

the application-layer activity remains similar. On the other hand, false positives, i.e., false alarms, can occur when non-dictionary attack flows end up with equal PPF values, enough to reach the threshold N.

B. Methodology

We perform the validation of this work by executing the state-of-the-art detection algorithm presented in [7] on the datasets listed in Section I. Instead of only considering the regular number of PPF in the detection algorithm, as would be the case in a regular flow monitoring setup, we also consider a compensated number of PPF. The compensated number of PPF consists of the number of the total number of packets metered for each flow minus the number of packets of all retransmissions and TCP control information fields. Ultimately, this should result in flat traffic when it comes to attacks.

By comparing the detection results when using non-compensated and compensated data, we can quantitatively evaluate the gain of 'flattening' traffic in the context of SSH intrusion detection. We perform the comparison in two dimensions – attacks and tuples – as this allows us to discover potential differences in the impact of compensation. Although comparing the number of detections in terms of attacks and tuples before and after compensation provides an indication of the detection improvements, it does not reveal anything about to accuracy of these detection outcomes. To assess these accuracies, we have performed a large-scale validation by collecting authentication logs of 58 machines on the campus network of the UT – 56 servers and 2 honeypots – to serve as the ground-truth for validation. These authentication logs are the only means of validating whether a machine has really been under attack. Since we only have the logs for UT hosts, we only consider the *UT* dataset in this part of the validation.

In the authentication logs, a minimum number of failed attempts must be encountered for the behavior to be considered a dictionary attack. Since the detection algorithm considers at least N consecutive flow records, only N or more connections to the SSH server that contain at least one failed attempt are considered. This comes down to at least five sessions with one or more authentication failures each. By evaluating log entries featuring this property, we created a list of attacks to serve as ground-truth for validation. This ground-truth can then be used for expressing the accuracy of the algorithm, both in terms of attacks and tuples, by comparing detection results to the ground-truth based on the following metrics:

- *True Positives* (TP) – Attacks/tuples correctly classified to feature a *brute-force* phase, for which 5 or more sessions with authentication failures are reported in the ground-truth.
- *False Positive* (FP) – Attacks/tuples incorrectly classified to feature a *brute-force* phase, for which less than 5 sessions with authentication failures are reported in the ground-truth.
- *True Negatives* (TN) – Attacks/tuples correctly classified to not feature a *brute-force* phase, for which less than 5 sessions with authentication failures are reported in the ground-truth.
- *False Negatives* (FN) – Attacks/tuples incorrectly classified to not feature a *brute-force* phase, for which 5 or more sessions with authentication failures are reported in the ground-truth.

Using these metrics, we can evaluate the differences in the detection algorithm for the non-compensated and compensated cases in terms of accuracy (Acc), which is defined as follows:

$$Acc = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \qquad (1)$$

In addition, to understand the relation between TCP control information and retransmissions, and geographical locations, we determine the physical origin of attacks and tuples based on a snapshot of the MaxMind GeoIP[4] database at the time of the measurements. The physical location can reveal why certain attacks or the majority of tuples are more likely to be detected only after compensation, as we hypothesize that retransmissions are strongly bound to the geographical distance between attackers and targets.

C. Results

The best way to visualize the achievements of this work is by means of a plot, as shown in Fig. 4. This figure shows the traffic in terms of the number of PPF over time between a single tuple of attacker and target. Clearly, the original network traffic (i.e., the sum of the three series in the figure) is not flat, but after compensating for control information packets and retransmissions, traffic that is almost flat remains. Occasional variations in the remaining number of PPF after compensation are the result of the performance trade-off discussed in Section III. We accept these variations, considering that most attacks feature a large enough number of flows to reach the threshold N.

The results of operating the detection algorithm on the considered datasets, both with and without PPF compensation, are shown in Table VII for attacks and Table IX for tuples. The number of detected attacks and tuples is considerably higher after compensation for both datasets. In *CESNET*, the total number of detected attacks is about a fourth times higher after compensation, i.e., from 9475 to 11849, while the improvement in terms of tuples is at 40%. For the *UT* dataset, a gain of 35% in terms of attacks can be observed – from 3499 to 4707 – and a gain of 45% in terms of tuples. The reason for the improvement in terms of attacks is that without compensation, the effects of retransmissions and control information hinder the detection for all tuples of an attack and, as such, the corresponding attack itself is also not detected.

Since we assume that retransmissions depend in part on the geographical location of and route between attacker and target, we show for each dataset the five countries from which

[4]We have used MaxMind's *GeoLite City* database, which can be retrieved from http://dev.maxmind.com/geoip/legacy/geolite/.

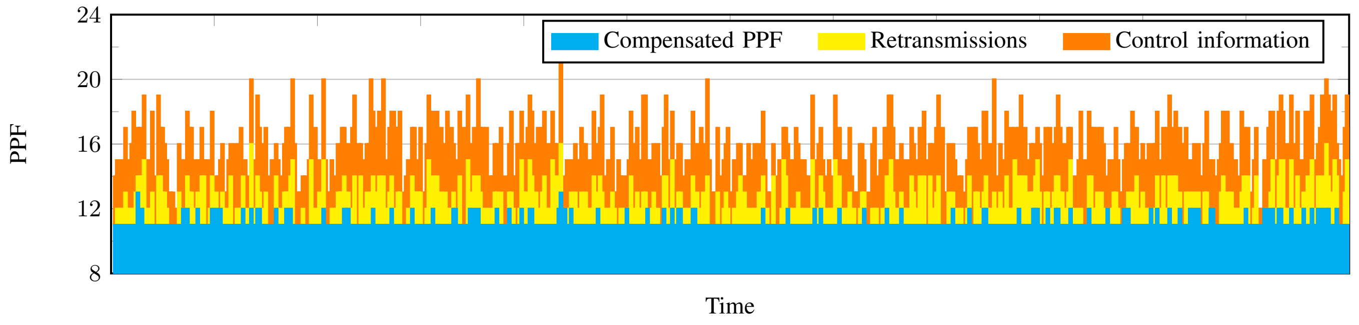

Fig. 4. Compensated brute-force flow records.

TABLE VII
Top five attack origins – Attacks

Dataset	Country	Non-compensated	Compensated
UT	China	1817	2347 (+29%)
	Netherlands	317	694 (+119%)
	Venezuela	195	233 (+19%)
	Russian Federation	165	189 (+15%)
	Chile	154	164 (+6%)
	Other	851	1080 (+27%)
	Total	**3499**	**4707 (+35%)**
CESNET	China	15239	19683 (+29%)
	United States	316	(+14%)
	Brazil	239	257 (+8%)
	Korea	146	170 (+17%)
	Turkey	124	139 (+12%)
	Other	1075	1420 (+32%)
	Total	**17139**	**21985 (+28%)**

TABLE IX
Top five attack origins – Tuples

Dataset	Country	Non-compensated	Compensated
UT	China	31887	55218 (+73%)
	Netherlands	11048	11646 (+5%)
	United States	3573	4203 (+18%)
	Vietnam	2358	2396 (+2%)
	Germany	1592	1642 (+3%)
	Other	10197	12939 (+27%)
	Total	**60655**	**88044 (+45%)**
CESNET	China	799840	1109458 (+39%)
	United States	37161	41230 (+11%)
	France	16096	22818 (+42%)
	Korea	10051	10890 (+8%)
	Malaysia	5579	5811 (+4%)
	Other	36994	48521 (+31%)
	Total	**905721**	**1234659 (+36%)**

TABLE VIII
Detection performance – attacks

Dataset	Logged attacks	TPR	FPR	TNR	FNR	*Acc*
UT	812	0.644	0.087	0.913	0.356	0.788
compensated		0.784	0.096	0.904	0.216	0.849

TABLE X
Detection performance – tuples

Dataset	Logged tuples	TPR	FPR	TNR	FNR	*Acc*
UT	4562	0.430	0.081	0.919	0.570	0.689
compensated		0.585	0.090	0.910	0.415	0.758

most attacks originate, both in terms of attacks (Table VII) and tuples (Table IX). The total number of countries involved in attacks is 60 for the *UT* dataset, and 71 for *CESNET*. Furthermore, we show the number of attacks and tuples reported only after compensation for those countries. Several observations can be made from the results. First, regarding the *UT* dataset, many attacks that are detected only after compensation have the attacking host located in China, with a figure of 530 attacks and 23331 tuples. While China easily outperforms the other countries in terms of attacks and tuples in *UT*, the relative increase of the number of attacks and tuples not reported until after compensation from China is also relatively high. More specifically, the increase in the number of attacks from China is 29%, and for tuples the increase is a staggering 73%. China also dominates in the *CESNET* dataset, where 4444 attacks from China are detected only after compensation, and 309618 tuples. The respective gains are 29% and 39%. Second, for the *UT* dataset, we implicitly know the geographical location of the targets of attacks. Moreover, we know that traffic between hosts located in China and the UT campus network is often susceptible to packet loss. The same can be said for the United States, for which an 18% gain in terms of tuples can be observed. All these observations make us conclude that TCP control information and retransmissions are indeed strongly bound to the distance in geographical location between attacker and target, and that the effects on detection can be observed quantitatively.

Out of the top five attack origins in *UT*, the gain in the number of detected attacks from The Netherlands after compensation is at 119%. This gain is higher than the 29% for China, for example, while attackers in The Netherlands are located closer (from a geographical point-of-view) to UT's campus network. Investigation of the measurement data has shown that for attacks where the route between attacker and target is not impaired by apparent packet loss or high(er) latencies, the non-flatness of network traffic is caused mostly by packets that relate to TCP's *delayed ACK* mechanism and not by retransmissions, which is more likely for far-away countries.

Thus far we have shown the different detection results when using compensated and non-compensated data. However, we have yet to compare these detection results to our ground-truth, consisting of authentication logs from 58 machines on the campus network of the UT. Since the ground-truth covers only a subset of the machines considered before, the number of attacks and tuples reported in the remainder of this section is lower than reported in Table VII and Table IX.

The detection performance of the detection algorithm in terms of attacks is shown in Table VIII, where we again divide the results in both compensated and non-compensated. Analogously, the detection performance in terms of tuples in shown in Table X. In both tables, we use the percentages of the previously introduced evaluation metrics. For example, the *True Positive Rate* (TPR) is the percentage of correctly identified attacks/tuples for which 5 or more sessions with authentication failures are reported in the ground-truth. The overall conclusion of the results is that compensation of the number of PPF yields a significantly improved TPR for both attacks and tuples. The TPR for attacks has improved from 64% to 78% for the *UT* dataset. For tuples, the figures are from 43% to 59%. These major improvements come at a minor cost in terms of false detections of roughly 1%. Also the accuracies for both attacks and tuples have improved significantly, from 79% to 85%, and 69% to 76%, respectively. We believe that the slight increase in the number of false detections is the case because of misclassified packets (e.g., unrecognized retransmissions), which in some cases cause benign network traffic to mimic dictionary attacks by becoming flat. These false positives are thus coupled to the performance trade-offs made in the plugin, as explained in Section III.

VI. Conclusions

In this work, we have measured the impact of TCP control information and retransmissions on networks in general, and on flow-based intrusion detection in particular. We set out by hypothesizing that the presence of these TCP phenomena in network traffic can impair analysis applications. Our measurements have confirmed that these phenomena are indeed omnipresent in any network. In our measurements, retransmissions account for 1–4% of all TCP packets, while control information accounts for significantly more, namely 27–33%. When it comes to bytes, each phenomenon contributes roughly 1–3% of all TCP traffic.

In the context of intrusion detection, we analyzed the impact of compensating for the considered TCP phenomena in a state-of-the-art dictionary attack detection algorithm for SSH. The results unmistakably demonstrate that flow-based intrusion detection benefits from a compensated number of PPF; the TPRs of the considered detection algorithm have improved for attacks and tuples from 64% to 78%, and from 43% to 59%, respectively. Moreover, the detection accuracies increase from 79% to 85%, and 69% to 76%. This was achieved without any change to the algorithm, i.e., in a manner transparent to the analysis application. This leads us to conclude that many of the possible flow monitoring applications mentioned in [4] can benefit from this work. From talks with a vendor of flow export devices we have even learned that adding a selection of the statistics used in this work to flow data is of interest to many customers, and that efforts are being undertaken to do so in one of their products.

As future work we consider analyzing variability at the application level by looking at protocols such as SSH. Variability in application-layer protocols may be introduced both intentionally and unintentionally, and can result from variable length application content or from random padding if the protocol in question allows it. Especially in case of intentional variation, hardly two flows in an attack may appear similar. Detecting the described forms of variability may aid in obtaining even flatter traffic patterns.

Acknowledgments

Special thanks go to Václav Bartoš (CESNET) for his valuable contributions to the research process, and the INVEA-TECH Community Program for providing parts of the measurement infrastructure used in this work. This work was partly funded by FLAMINGO, a Network of Excellence project (ICT-318488), and SALUS, a STREP project (ICT-313296), both supported by the European Commission under its Seventh Framework Programme.

References

[1] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with Netflow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[2] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011 (Internet Standard), Internet Engineering Task Force, September 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7011.txt

[3] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Memo), Internet Engineering Task Force, October 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[4] T. Zseby, E. Boschi, N. Brownlee, and B. Claise, "IP Flow Information Export (IPFIX) Applicability," RFC 5472 (Internet Standard), Internet Engineering Task Force, September 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5472.txt

[5] R. Hofstede, I. Drago, A. Sperotto, R. Sadre, and A. Pras, "Measurement Artifacts in NetFlow Data," in *Proceedings of the 14th International Conference on Passive and Active Measurement, PAM'13*, ser. Lecture Notes in Computer Science, vol. 7799. Springer Berlin Heidelberg, 2013, pp. 1–10.

[6] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A Flow-Based SSH Intrusion Detection System," in *Proceedings of the 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS'12*, ser. Lecture Notes in Computer Science, vol. 7279. Springer Berlin Heidelberg, 2012, pp. 86–97.

[7] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "SSH Compromise Detection using NetFlow/IPFIX," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 20–26, 2014.

[8] J. Vykopal, “Flow-based Brute-force Attack Detection in Large and High-speed Networks,” Ph.D. dissertation, Masaryk University, 2013.

[9] M. Drašar, “Protocol-Independent Detection of Dictionary Attacks,” in *Proceedings of the 19th EUNICE Open European Summer School, EUNICE'13*, 2013, pp. 304–309.

[10] M. Vizváry and J. Vykopal, “Flow-based detection of RDP brute-force attacks,” in *Proceedings of 7th International Conference on Security and Protection of Information, SPI'13*, 2013, pp. 131–138.

[11] A. Satoh, Y. Nakamura, and T. Ikenaga, “A flow-based detection method for stealthy dictionary attacks against Secure Shell,” *Journal of Information Security and Applications*, 2014.

[12] J.-C. Bolot, “End-to-End Packet Delay and Loss Behavior in the Internet,” *ACM SIGCOMM Computer Communication Review*, vol. 23, no. 4, pp. 289–298, 1993.

[13] T. Lakshman and U. Madhow, “The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss,” *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336–350, 1997.

[14] R. Braden, “Requirements for Internet Hosts – Communication Layers,” RFC 1122 (Internet Standard), Internet Engineering Task Force, October 1989. [Online]. Available: http://www.ietf.org/rfc/rfc1122.txt