Flow-Based Compromise Detection:

Lessons Learned

Rick Hofstede, Aiko Pras, and Anna Sperotto | University of Twente
Gabi Dreo Rodosek | Universität der Bundeswehr München

Although the aggregated nature of exported flow data provides many advantages in terms of privacy and scalability, flow data may contain artifacts that impair data analysis. In this article, we investigate the differences between flow data analysis in theory and practice—that is, in lab environments and production networks.

Brute-force attacks are as old as the Internet, and therefore antiquated, yet their popularity is increasing.[1,2] One of the main targeted services of these attacks is Secure Shell (SSH). In 2015, the threat intelligence organization Talos, together with Tier 1 network operator Level 3 Communications, mitigated SSH brute-force attacks from a group named SSHPsychos or Group 93, generating more than 35 percent of the global SSH network traffic.[3]

Another target of brute-force attacks is rapidly gaining popularity: content management systems (CMSs), such as WordPress. For example, failed authentication attempts on WordPress instances behind the security company Sucuri's protection services showed an increase of a factor eight over a period of six months in 2015.[2]

While brute-force attacks are omnipresent and their existence widely known, we know that only few attacks actually result in compromises.[4] We therefore believe that monitoring would be more effective once it targets compromises rather than attacks.

Monitoring compromises is vital for maintaining secure networks. Traditionally, security monitoring is performed in a host-based fashion by running intrusion detection systems (IDSs) on networked devices. As such, IDSs have access to network interfaces and file systems, allowing them to achieve high detection rates with few false positives and negatives. However, the problem with host-based approaches is their scalability. In environments where IT service departments do not have global machine access, it is virtually impossible to install and manage agents on every machine. For this reason, network-based approaches may be used for monitoring networked systems, where information is gathered at central observation points within the network.

One approach is to capture individual packets. However, this results in large amounts of data. Especially when packet payloads are captured, the scalability of packet-based approaches is limited due to stringent hardware requirements for storage and analysis. It should be noted, however, that due to the ever-increasing amount of network traffic being encrypted, the benefit of storing packet payloads is vanishing.

Another, more scalable, network-based monitoring approach is to analyze traffic flows using export capabilities of packet-forwarding devices or dedicated appliances (probes).[5] These flow exporters aggregate packets into flows and export flow data using protocols like NetFlow and IPFIX.

Flow data is a proven source of information for detecting various types of security incidents, such as

 Copublished by the IEEE Computer and Reliability Societies

distributed denial-of-service (DDoS) attacks and network scans.[5] However, for analyses like SSH and web application compromise detection, the use of flow data is much harder. First, the lower granularity of flow data (compared to packet-based alternatives) generally results in more false positives and negatives. Second, artifacts in flow data, such as inaccuracies and packet loss, can be subject to misinterpretation. It is often underestimated how much flow data in theory and lab environments may differ from flow data in practice, rendering many flow-based analyses unsuitable for production usage. In this article, we investigate whether flow-based compromise detection for SSH and web applications is possible in production networks, or whether its application is limited to lab environments. The work presented here is a summary of four years of PhD research.[6]

Compromise Detection—A Case Study

Compromised machines are the core building blocks of many illegal activities on the Internet. Examples of such activities are spamming, DDoS attacks, and illegal content distribution. Detecting and quarantining compromised machines is therefore of vital importance for maintaining secure networks.

Compared to host-based IDSs, network-based IDSs are far behind when it comes to compromise detection. They generally report on the presence of attacks, regardless of whether the attacks were successful or not. Again, high false positive rates and the presence of data artifacts have prevented advanced flow-based security analyses in general and compromise detection in particular from breaking through.

A typical application for compromise detection is SSH. With almost 26 million connected and scannable SSH daemons in November 2015, according to Shodan (www.shodan.io), SSH daemons are a popular attack target.[4] SSH is used for remote server administration; therefore, the compromise of a target machine immediately results in adversaries gaining unprivileged control. In this section, we describe our experiences with SSH compromise detection.

SSH Compromise Detection in Theory

Brute-force attacks aim to compromise user accounts by trying many combinations of usernames and passwords. One particular type of brute-force attack is the *dictionary attack*. Dictionaries are lists of frequently used username and password combinations. Attacks using dictionaries are particularly effective because purely random passwords are difficult to remember and therefore less common than passwords that are simple and easy to remember.

Before a target can be attacked, it must be discovered. This can be done by scanning a network, for example, by using tools like nmap (nmap.org) or by using publicly available lists of potential targets, such as those provided on PasteBin (pastebin.com) or gathered by services like Shodan. Once potential targets have been found, brute-force attacks can be launched. Eventually, targets may be compromised, depending on whether valid credentials were discovered. In short, we can say that brute-force attacks consist of three phases: a *scan* phase, followed by a *brute-force* phase, potentially concluded by a *compromise* phase if the attack was successful.

Although these attack phases feel rather natural, they were formalized first in the context of network flows in 2009 in "Hidden Markov Model Modeling of SSH Brute-Force Attacks."[7] It was found that they could be easily identified in lab environments by the number of packets per flow (PPF).

For example, the *scan* phase features a low number of PPF since it consists of only one or two TCP SYN packets, while the *brute-force* phase is characterized by a significantly higher number of PPF, for instance, 10 to 15, due to the SSH connection initiation and one or more authentication attempts. In theory, traffic in the *brute-force* phase is flat, that is, alike in terms of packets, bytes, and duration, because of repeated application-layer actions or events, namely authentication attempts. In a compromise, we may observe either a number of PPF that is higher than the *brute-force* phase's, if the target is being actively misused, or lower, if the connection to the target is maintained but left idle. The behavior of a typical SSH brute-force attack in terms of the number of PPF is shown and summarized in Figure 1.

SSH Compromise Detection in Practice

The theoretical model described in "Hidden Markov Model Modeling of SSH Brute-Force Attacks"[7] aimed at defining brute-force attack behavior, including the *scan*, *brute-force*, and *compromise* phases. To demonstrate that flow-based compromise detection is feasible, we developed an open-source application named SSHCure (https://github.com/sshcure/sshcure). SSHCure has a strong focus on detecting the three attack phases and was the first flow-based IDS that could report on compromises. The detection was realized by monitoring the number of PPF between a potential pair of attacker and target to identify flat traffic. If the traffic was found to be flat enough, a *brute-force* phase was detected, and any significant change to the traffic flatness was thought to be indicative of a compromise.

Since SSHCure relies on flow data exported using NetFlow or IPFIX, it was believed to work on flow data from any source, like any other flow data analysis software. To validate this, we tested SSHCure on our own

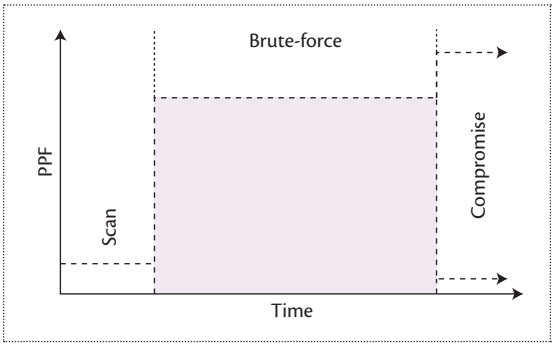

Figure 1. Theoretical brute-force attack behavior.

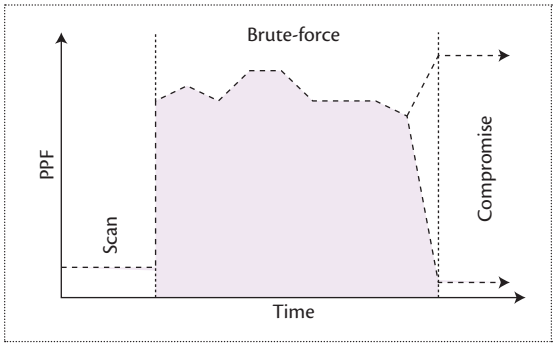

Figure 2. Brute-force attack behavior under network and measurement artifacts.

campus network as well as on a backbone link of the Czech NREN (CESNET). In addition, we promoted SSHCure at many conferences and gave demos at various companies. The result was wide user base, ranging from small web-hosting companies to backbone networks and governmental CSIRTs. However, as soon as SSHCure was released and after we requested feedback from users, we received mixed results. For some, SSHCure worked great and provided them with a powerful tool to detect compromises. Others, however, were reporting on false positive and negative detections.

Analysis of the reported problems revealed that the vast majority was caused by a broken assumption; although flat traffic was thought to signify brute-force traffic,[8,9] we discovered this was often not the case.[10] So instead of the theoretical attack behavior, depicted in Figure 1, we discovered that attacks may feature deviations in the number of PPF, as shown in Figure 2, for network and measurement artifacts.

Another source of false positive compromise detections was authentication monitors, such as Fail2ban (www.fail2ban.org) and DenyHosts (denyhosts.sourceforge.net), which block connecting hosts after a number of failed login attempts. When blocking a connecting host after a *brute-force* phase, the result would be connection initiations toward the target—on the network-level, those initiations appear exactly like the behavior of a compromise in Figures 1 and 2.

Lessons Learned

Our case study has shown that we cannot just assume flow data in production networks to be similar, in terms of quality, to flow data in lab environments. Flow data analysis is therefore not as simple as it is often assumed. In the upcoming sections, we discuss why flow data analysis is sometimes impaired. We found that two kinds of artifacts may severely impair analysis: measurement artifacts and network artifacts (see Figure 3).

Measurement Artifacts

Measurement artifacts cause traffic metadata to not fully resemble the original traffic anymore, and therefore likely impair data analyses. These artifacts may range from data loss as a consequence of under-dimensioned network links and storage to complex race conditions in the firmware of export devices. Since flow export is widely supported by higher-end packet-forwarding devices and can typically be enabled very easily, we initially assumed the presence of network measurement artifacts to be limited. After we discovered that measurement artifacts were not uncommon in flow data, we systematically compared six frequently used flow export devices for the presence of measurement artifacts.[11] Other works on measurement artifacts in flow data have appeared over the years as well, such as "Peeling Away Timing Error in NetFlow Data,"[12] reporting on timing issues, and "Uncovering Artifacts of Flow Measurement Tools,"[13] reporting on artifacts in flow data from widespread Juniper devices. We elaborate on the most important artifacts that we found to impair flow-based compromise detection. We illustrate each artifact by means of an example in the context of SSH. Unless indicated differently, the examples show traffic from attacker to target, that is, traffic with destination port 22. Since understanding this requires technical knowledge on flow monitoring, we refer the reader to "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX"[5] for a comprehensive tutorial on the matter.

TCP Flows without Flags

TCP flags in flow data are an invaluable source of information when it comes to understanding network behavior. Their availability is however far less common than it seems; many (older) flow exporters, especially those embedded in packet-forwarding devices that perform forwarding in hardware, do not export TCP flags for hardware-switched flows. (Higher-end packet-forwarding devices typically feature [fast]

hardware and [slow] software paths. Most packets should be switched in hardware to achieve the highest performance.) In a typical campus scenario, this corresponds to roughly 99.6 percent of all TCP flows.[11]

An exemplary situation in which the lack of TCP flags can easily lead to erroneous conclusions is the following. Although rather unknown, the OpenSSH SSH daemon includes functionality for rate-limiting connection requests. On the network-level, this functionality appears similar to authentication monitors like Fail2ban and DenyHosts when blocking connecting machines, due to a source or attacker vainly trying to establish TCP connections to the target. Without TCP flags, the resulting flow data is in line with the following:

ID	Start	End	Src. port	Flags	Packets
1	10:37:09	10:37:13	37162	.....	13
2	10:37:17	10:37:22	37165	.....	13
3	10:37:29	10:37:33	37193	.....	3

Several conclusions could be drawn:

- Flow record 3 represents a newly established connection and signifies a compromise, that is, featuring TCP SYN, ACK, and PSH flags.
- Flow record 3 is part of a long connection and may signal a slow *brute-force* phase or a compromise, that is, featuring only TCP ACK and PSH flags.
- Flow record 3 represents three connection attempts to the daemon, while the target blocks the attacker. In this case, the flow record would feature merely a TCP SYN flag.

Most important is that regardless of the conclusion drawn, the illustrated behavior appears very similar to what is shown in Figure 1. Several *brute-force* phase flows can be observed, followed by a flow that has a significantly different number of PPF (3) than the flows in the *brute-force* phase—a potential sign of a compromise. TCP flags for the last flow would however reveal that this is certainly not indicating a compromise. Since a successful TCP three-way handshake and subsequent SSH connection setup would require at least multiple TCP ACK flags, observing only the TCP SYN flag indicates that a connection was never established.

Imprecise Flow Cache Entry Expiration

Flow exporters maintain a flow cache for accounting active flows. Once a flow is considered to have terminated, cache entries are expired and removed from the cache, and inserted in NetFlow or IPFIX messages. Although expiring and removing flow cache entries should theoretically be done (almost) simultaneously,

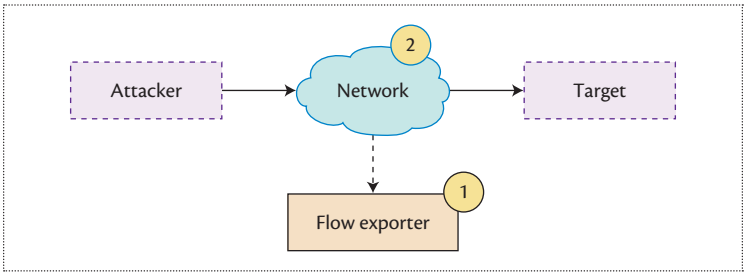

Figure 3. Components where artifacts may be introduced during brute-force attacks.

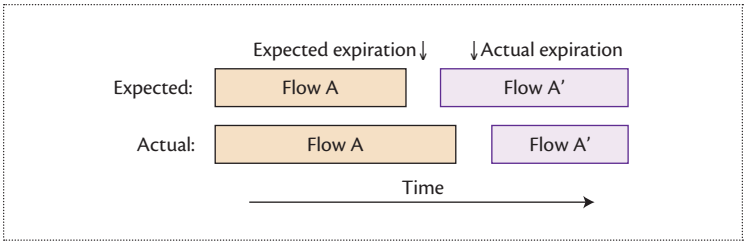

Figure 4. Accidental merging of flows due to imprecise flow cache entry expiration.

measurements presented in "Measurement Artifacts in NetFlow Data"[11] have shown that this is often not the case. The consequence: two or more connections that are mapped to the same cache entry are (partially) merged, as shown in Figure 4.

The consequences of imprecise flow cache entry expiration can best be explained by means of Figure 5. The figure shows an attack in the *brute-force* phase that features a small spike in terms of the number of PPF, meaning that there was a single flow that featured more packets than the other flows. Intuitively, one may believe that this is a compromise. The fact that an authentication attempt was successful and perhaps the attacker executed some commands is likely to feature a different number of PPF than the other flows in the *brute-force* phase. In early versions of SSHCure, this event would be reported as a compromise.

Understanding whether a spike in flat traffic is caused by application behavior or measurement artifacts is far from trivial. However, repairing or working around this artifact is often possible, but only if TCP flags are available. This can be illustrated by means of the following example. If flow cache entry expiration works well, two flows that map to the same flow cache entry are being exported in two different flow records:

ID	Start	End	Src. port	Flags	Packets
1	10:37:09	10:37:13	37162	.AP.SF	13
2	10:37:17	10:37:22	37162	.AP.SF	13

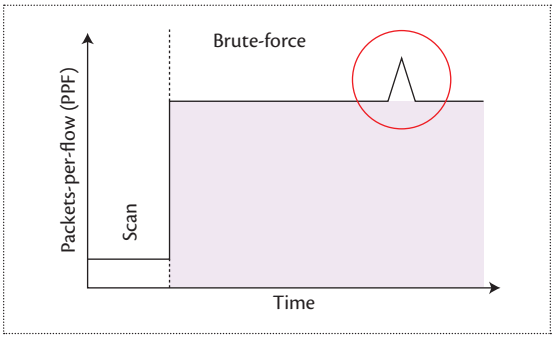

Figure 5. Deviation in the *brute-force* phase: compromise or artifact?

However, if cache entries are not expired and exported precisely, the following may happen:

ID	Start	End	Src. port	Flags	Packets
3	10:37:09	10:37:16	37162	.AP.SF	16
4	10:37:16	10:37:22	37162	.AP..F	10

The first three packets of the second flow are merged with the first flow, causing a sudden spike (16 PPF instead of 13).

Although resource intensive, it might be possible to detect this case by not flagging the compromise until flow record 4 is detected within reasonable time. This would allow the analysis to conclude that the sum of packets of records 3 and 4 equals the (expected) sum of packets of records 1 and 2, and that erroneous merging may have occurred.

There can however be situations where the result of imprecise expiration is even worse:

ID	Start	End	Src. port	Flags	Packets
5	10:37:09	10:37:22	37162	.AP.SF	26

Although in this case it may seem likely that two flows were merged erroneously (because the number of PPF equals the sum of packets of records 1 and 2), we can only guess whether this is a case of two merged flows, or a compromise.

The extent that imprecise flow cache entry expiration affects the exported flow data depends strongly on the design and specifications of the export device, and the traffic load to which the export device is exposed. We therefore cannot provide any general numbers on this artifact, but refer the reader to "Measurement Artifacts in NetFlow Data"[11] for a detailed analysis.

Gaps

Gaps can be found in flow data both on the level of packets and flows. In any case, they signal data loss and are therefore one of the worst types of artifacts. Note that gaps may also be introduced intentionally, typically to reduce load of monitoring devices, which is commonly referred to as *sampling*. In this article, we consider only unintended gaps, leaving sampling out of scope.

When not all packets of a flow are metered, a flow exporter may be suffering from high load. Although missing packets in a flow can be catastrophic for analysis, they can often be compensated for using clever techniques. This is illustrated by the following example:

ID	Start	End	Src. port	Flags	Packets
1	10:31:53	10:31:57	37008	.AP.SF	13
...					
2	10:37:09	10:37:13	42815	.AP.SF	13
3	10:37:27	10:37:31	37008	.AP..F	12

For flow record 1, we can be rather sure that all packets have been metered. At least, we have observed all expected TCP flags (that is, TCP SYN, ACK, PSH, and FIN), meaning that there has been some application-layer information exchange between attacker and target. Later in time, we observe a flow (ID 3) with the same source port that features one packet less and no TCP SYN packet. Given that the flow represented by record 1 has completed, as we observed a TCP FIN flag, and there have been no other flows with the same source port between these two flows, we can assume that the non-metered TCP SYN packet is likely a measurement artifact. Also, it should be noted that many stateful middleboxes would likely have dropped the flow with ID 3 if the TCP SYN packet was really missing. A more severe type of gap is when flows are not being accounted at all, which happens especially in situations where the flow cache is under-dimensioned and full. When this "lost" flow happens to feature a compromise, the IDS will never be able to detect it.

The problem of under-dimensioned flow caches can usually be observed only in flow exporters with hardware caches. Common examples are the Cisco Catalyst 6500 and 7600 series routers, Cisco's most-sold routers for campus environments.[14] Older versions of these devices come with fixed-size caches of 128,000 or 256,000 entries, which is rather small for typical university deployments without sampling enabled.[11] Exporters that are fully built-in software, however, often use constructions like linked lists for handling cases in which the cache reaches its maximum utilization, although that may reduce overall exporter performance.

Some gaps in flow data may be detected by inspecting the sequence numbers in NetFlow and IPFIX protocol messages. In the case of NetFlow v5 and IPFIX, the sequence number records the number of exported flow records, while for NetFlow v9, sequence numbers record the number of protocol messages. Gaps that are a consequence of exporter overload are, however, unlikely

to be recorded in the protocol's sequence numbers and hence harder to detect. Also, given that gaps in flow data are completely load dependent, it is hardly possible to provide general numbers on the presence of this artifact.

Timestamp Errors

NetFlow and timing problems are widely known to go hand in hand. For example, clock resolutions of various flow exporters are described in "One-Way Delay Measurement Based on Flow Data: Quantification and Compensation of Errors by Exporter Profiling."[15] The authors conclude that "timestamp errors in flow data are dominated by errors resulting from limited timestamp resolution." Several other related problems are described and quantified in "Peeling Away Timing Error in NetFlow Data."[12] First, the authors describe a cyclic error of up to one second that is inherent to the design of NetFlow v9. Second, due to the implementation of typical NetFlow metering processes, timestamps may be off by several seconds. The main conclusion of the authors is that "any assumption that devices exporting NetFlow v9 are capable of millisecond-level accuracy and/or strict ordering of flows does not hold."

An illustrative example of a timestamping artifact was discovered during the development of SSHCure and involves the Cisco Catalyst 6500/SUP2T. (The artifact was observed in Cisco IOS v15.1(1)SY and v15.1(1)SY3.) This artifact causes the start time of a few flows to be set to a time that comes even before the boot time of the forwarding device, which can impair subsequent analysis, such as SSHCure's. Since SSHCure is developed as a plugin for NfSen, which uses flow data chunks of five minutes, many functions operate once every five minutes. However, to improve the accuracy of internal components, we decided to divide chunks into sub-chunks of one minute. The only way to do this is by dividing the interval between the first and last flow record of every chunk into equally sized sub-chunks. However, when some timestamps feature major deviations, this leads to major problems, since it will cause the set of flow records to not be distributed uniformly over the sub-chunks. Subsequently, thresholds are not calculated correctly, resulting in false positives and negatives.

Lessons Learned

Artifacts exist in practically any measurement device, even in devices from renowned vendors, and we have highlighted artifacts of various severities. The wide presence of artifacts in flow exporters almost seems paradoxical, since flow export is often presented as "plug-and-play" functionality. According to vendors, merely enabling the functionality brings all the nice features of flow export technologies, while the exported data should actually be treated with care. However, it should be clear that every application has its own requirements on data quality.

Network Artifacts

Solid measurements require a calibrated measurement infrastructure. But even in environments with an optimal measurement infrastructure, problems may arise. In this section, we discuss a second class of artifacts, namely network artifacts. The presented artifacts cannot be discriminated in flow data and affect the flatness of brute-force traffic, effectively impairing compromise detection. For a comprehensive overview and discussion on the presented artifacts, we refer the reader to "Unveiling Flat Traffic on the Internet: An SSH Attack Case Study."[10]

TCP Retransmissions

After SSHCure was released to the public, several users reported on false positive compromise detections. This came as a surprise, since we validated SSHCure's detection performance in our campus network based on a large-scale validation involving almost 100 machines over a period of two months.[4] Users started to investigate authentication logs of machines that were reported to be compromised, yet did not find any signs of compromises. To rule out the presence of measurement artifacts as described earlier, users were asked to provide us with packet traces, which provided us clear insights into the problem. Retransmissions were affecting attacks from countries that are far away from the observation point, such as the Far East in the case of observation points in Europe.

Retransmissions are the cornerstone of reliable communication channels. Despite the ever-increasing bandwidths toward end systems, TCP retransmissions occur at all times and feature clear diurnal patterns. In a recent work, we have shown that in an academic campus network, retransmissions account for 1.43 and 0.97 percent of all packets and bytes, respectively.[10] Analogously, we have measured 3.22 percent of all packets and 1.22 percent of all bytes on a link to the "commercial Internet" of an academic backbone.

The problem with retransmissions and flow data is that flow export devices generally count packets and bytes at the network layer (L3), while retransmissions can only be discriminated at the transport layer (L4). As such, retransmissions cannot be discriminated from other packets, directly affecting the detection metric used by SSHCure, that is, the number of PPF. In case of retransmissions, spikes can be observed when analyzing brute-force behavior over time, as shown in Figure 5.

TCP Control Information

TCP features many mechanisms for connection probing and optimizing traffic flow between endpoints. Most fall under the umbrella of TCP control information, among

which are duplicate and non-piggybacked acknowledgments, window updates, and keep-alive probes. Contrary to retransmissions, which result in additional packets in the network by definition, control information often employs TCP header fields and therefore does not necessarily result in additional packets. However, in some situations, it may consume additional bandwidth.

An exemplary type of control information that affects the amount of network traffic exchanged between two endpoints is TCP's delayed acknowledgment mechanism. Its goal is to slightly delay the transmission of acknowledgments, such that the successful reception of multiple packets can be acknowledged at once. Whether additional packets/acknowledgments are sent heavily depends on timing, making it a dynamic mechanism that is not constantly activated. The work described in "Unveiling Flat Traffic on the Internet: An SSH Attack Case Study"[10] shows that TCP control information can be observed at any time of the day, in both academic and "commercial" networks, and between endpoints both close to and far away from the observation point.

The effects of TCP control information on brute-force traffic flatness should be clear. The unpredictable nature of some sorts of control information (for example, due to timing) makes assumptions on traffic flatness to fail. The measurements presented in "Unveiling Flat Traffic on the Internet: An SSH Attack Case Study"[10] show that approximately 30 percent of all TCP packets in the monitored networks carry control information, which clearly underlines the importance of considering this type of traffic in detection algorithms.

Lessons Learned

In this section, we described two network artifacts that cause sudden deviations in network traffic, in particular when looking at the number of PPF over time. Traditionally, there is no way to discriminate retransmissions and control information packets from other packets in flow data. In "Unveiling Flat Traffic on the Internet: An SSH Attack Case Study,"[10] however, we have shown how to overcome this problem. By exporting the number and size of TCP retransmissions and control information packets on a per-flow basis, the "real" number of packet and bytes can be measured. In addition, we have measured the impact of the compensation measures taken. For example, without any change to the detection algorithm, the accuracy of state-of-the-art IDSs improved significantly by more than 6 percentage points.

Discussion

The previous sections have shown that flow-based techniques can be used to determine which hosts are compromised after brute-force SSH dictionary attacks. But can flow-based techniques also be used to detect compromises of other Internet services, such as web applications? To answer this question, we conducted a number of experiments on CMSs, since such systems are often vulnerable to brute-force attacks and, once hacked, may be misused for spam campaigns or DDoS attacks. Such hacked CMSs may lead to unexpected side effects, such as the entire IP address space of the web-hosting company being blacklisted. Hosting companies therefore often monitor webserver logfiles of all virtual web hosts, but in cases where virtual private servers (VPSs) are provided to customers, logfile analyses may be impossible. In such cases, flow-based techniques may be an interesting alternative to detect CMS compromises.

Our experiments were conducted on the network of Hosting 2GO, a Dutch Top-10 web-hosting provider, over a period of one month in the summer of 2015. Roughly 2,500 hosts were monitored and more than 400 GB of flow data analyzed. To validate our approach, results were compared to server logfiles, which were considered to represent ground truth. Similar to the SSH case, we started with an elementary approach based on attack signatures. The main lesson learned from that experiment was that simple signatures may provide a basis for detection, but the number of false positives is too high due to many small connections, such as traffic generated by web crawlers, calendar fetchers, and photo galleries. We therefore adapted our approach; instead of focusing on sharp transitions between the *brute-force* and the *compromise* phase (see Figure 2), we created per-connection histograms that provide information on packet payload sizes in flow data. Clustering techniques were subsequently applied to group attack traffic and benign traffic. Such histograms turned out to deliver good detection results, and can be seen as an alternative approach to overcome the problems introduced at the TCP level, like retransmissions and control information, as described earlier. Details of these experiments can be found in the PhD thesis that resulted from this work.[6]

We believe that security researchers should focus more on compromise detection, instead of traditional attack detection. We should not worry too much when we see failed attacks, but rather be alarmed whenever we see successful attacks (compromises). In this article, we wanted to investigate whether flow-based techniques to detect compromises could be applied in practice. We focused on two cases: SSH and web applications. From our research, we may conclude that flow-based techniques not only work in lab environments, but can also work in real operational environments. However, it is crucial that the measurement infrastructure is calibrated and network artifacts are understood. Our SSH compromise analysis has revealed several artifacts that would not appear in lab environments, such as imprecise expirations and gaps. Our web-based CMS analysis learned that elementary attack signatures resulted in too many

false positives, but by replacing such signatures by histograms and clustering techniques, good results were possible. Therefore, our general conclusion is that flow-based compromise detection is a promising technique, although additional research is still needed to cover more application scenarios and to better understand all forms of artifacts that may occur in operational environments. ■

References

1. "SSH Brute Force—The 10 Year Old Attack That Still Persists," Sucuri Inc., July 2015; https://blog.sucuri.net/2013/07/ssh-brute-force-the-10-year-old-attack-that-still-persists.html.
2. "WordPress Brute Force Attacks," Sucuri Inc., 2015; https://sucuri.net/security-reports/brute-force.
3. *Safeguarding the Internet*, tech. report, Level 3 Communications, June 2015; http://www.level3.com/;/media/files/white-paper/en_secur_wp_botnetresearchreport.pdf.
4. R. Hofstede et al., "SSH Compromise Detection Using NetFlow/IPFIX," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, 2014, pp. 20–26.
5. R. Hofstede et al., "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, 2014, pp. 2037–2064.
6. R. Hofstede, "Flow-Based Compromise Detection," PhD dissertation, University of Twente, Enschede, The Netherlands, 2016.
7. A. Sperotto et al., "Hidden Markov Model Modeling of SSH Brute-Force Attacks," Integrated Management of Systems, Services, Processes and People, *Proceedings of the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management* (DSOM 09), LNCS 5841, Springer Berlin Heidelberg, 2009, pp. 164–176.
8. J. Vykopal, "Flow-Based Brute-Force Attack Detection in Large and High-Speed Networks," PhD dissertation, Masaryk University, Brno, Czech Republic, 2013.
9. M. Drasar, "Behavioral Detection of Distributed Dictionary Attacks," PhD dissertation, Masaryk University, Brno, Czech Republic, 2015.
10. M. Jonker et al., "Unveiling Flat Traffic on the Internet: An SSH Attack Case Study," *Proceedings of the 14th IFIP/IEEE Symposium on Integrated Network and Service Management* (IM 15), 2015.
11. R. Hofstede et al., "Measurement Artifacts in NetFlow Data," *Proceedings of the 14th International Conference on Passive and Active Measurement* (PAM 13), LNCS 7799, Springer Berlin Heidelberg, 2013, pp. 1–10.
12. B. Trammell et al., "Peeling Away Timing Error in NetFlow Data," *Proceedings of the 12th International Conference on Passive and Active Measurement* (PAM 11), LNCS 6579, Springer Berlin Heidelberg, 2011, pp. 194–203.
13. I. Cunha et al., "Uncovering Artifacts of Flow Measurement Tools," *Proceedings of the 10th International Conference on Passive and Active Measurement* (PAM 09), LNCS 7799, Springer Berlin Heidelberg, 2009, pp. 187–196.
14. J.H. Follett, "Cisco: Catalyst 6500 The Most Successful Switch Ever," June 2006; http://www.crn.com/news/networking/189500982/cisco-catalyst-6500-the-most-successful-switch-ever.htm.
15. J. Kögel, "One-Way Delay Measurement Based on Flow Data: Quantification and Compensation of Errors by Exporter Profiling," *Proceedings of the International Conference on Information Networking* (ICOIN 11), 2011, pp. 25–30.

Rick Hofstede is a former PhD student of the University of Twente and the Universität der Bundeswehr München, Germany. In 2016, he successfully defended his PhD thesis titled "Flow-Based Compromise Detection," after which he switched to the cybersecurity industry. His main areas of interest include cybersecurity, big data processing, and network forensics. Contact him at r.j.hofstede@alumnus.utwente.nl.

Aiko Pras is professor at the University of Twente, the Netherlands, where he is member of the Design and Analysis of Communication Systems (DACS) group. His research interests include Internet security, measurements, and management. He is chairing the IFIP Technical Committee on Communications Systems (IFIP-TC6), and has been chair of the EU Future Internet cluster and coordinator of the European Network of Excellence on Management of the Future Internet (FLAMINGO). Contact him at a.pras@utwente.nl.

Anna Sperotto is assistant professor at the Design and Analysis of Communication Systems Group of the University of Twente, the Netherlands. She received a PhD degree from the University of Twente, in 2010, with the thesis titled "Flow-Based Intrusion Detection." Her research interests include network security, network measurements, and traffic monitoring and modeling. Contact her at a.sperotto@utwente.nl.

Gabi Dreo Rodosek holds the Chair for Communication Systems and Network Security at the Universität der Bundeswehr München, Germany. She is director of the research institute CODE (Cyber Defence), member of the Supervisory and Advisory Board of Giesecke & Devrient GmbH, member of the advisory board of BWI GmbH, member of the Governing Board of the Deutsches Forschungsnetz (DFN) and member of the IT expert panel of the German federal financial supervisory agency (BaFin). In 2016, she was awarded the European Medal from the Bavarian Minister for European Affairs and International Relations Dr. Beate Merk. Contact her at gabi.dreo@unibw.de.