# Flow-based Brute-force Attack Detection

Martin Drašar, Jan Vykopal
Masaryk University
Institute of Computer Science
Botanická 68a, 602 00 Brno, Czech Republic

Philipp Winter
Karlstad University
Department of Computer Science
Universitetsgatan 2, 651 88 Karlstad, Sweden

**Abstract**

Brute-force attacks are a prevalent phenomenon that is getting harder to successfully detect on a network level due to increasing volume and encryption of network traffic and growing ubiquity of high-speed networks. Although the research in this field advanced considerably, there still remain classes of attacks that are undetectable. In this chapter, we present several methods for the detection of brute-force attacks based on the analysis of network flows. We discuss their strengths and shortcomings as well as shortcomings of flow-based methods in general. We also demonstrate the fragility of some methods by introducing detection evasion techniques.

## 1 Introduction

In recent years, network security research started focusing on *flow-based* attack detection in addition to the well-established *payload-based* detection approach. Instead of only looking for malicious activity in the actual packet data, network flows are also considered for analysis [8]. This is not surprising since the amount of data one has to fight with is drastically reduced and the attacks visible in flow data tend to complement the attacks that we strive to find in network payload.

In this chapter, we give a compact overview about current research in this field with respect to brute-force attacks. We propose five detection techniques and shed light on the shortcomings inherent to the flow-based attack detection approach.

This chapter is divided into five sections. The rest of this section highlights the difference in attack orchestration and discusses detection in encrypted traffic. Section 2 describes five different approaches to flow-based intrusion detection that reveal brute-force attacks. Limitations imposed by the nature of flows are summarized in Section 3. Four detection evasion techniques are then outlined in Section 4. The chapter is recapitulated in Section 5. Flow data collection in large scale networks is extensively covered by Chapter

**TODO for editors: link to the chapter by Pavel Celeda and Vojtech Krmicek.**

## 1.1 Noisy Versus Stealthy Attacks

Attacks that occur in a network can be roughly divided into two categories, depending on their impact on traffic patterns. On the one side there are *noisy attacks* that disrupt these patterns significantly. One example is port scans that often precede actual attacks [9]. Such attacks are very easy to detect since all that is needed is to look for a sudden increase in traffic volume. Noisy attacks are useful to penetrate networks that are not sufficiently protected and to estimate defense capabilities of particular networks. They can also be used as a cover for stealthy attacks running simultaneously. Any exposed network is likely to be target sooner or later, so it is easy to gather real life examples.

On the other side, there are *stealthy attacks*. These attacks are much harder to gather and examine as they by virtue try to remain undetected. Stealthy attacks have to be crafted for a special target network and must reflect its detection capabilities. Staying under the radar also means that the attack is generally slower and that it has to run longer.

## 1.2 Detection of Attacks in Encrypted Traffic

Various secured protocols, services and applications became more and more popular in recent years. Besides services such as SSH, even web applications provided by Google or Facebook are currently accessible over HTTPS. Furthermore, user authentication via secured communication channels is becoming a standard these days.

With the rise of encrypted traffic, the traditional approach to network-based intrusion detection is becoming *ineffective*. Packet payload which is searched for signatures of known attacks by deep packet inspection is opaque, only packet headers can be analyzed. Therefore, flow-based detection is one of the possible ways to deal with encrypted traffic.

# 2 Detection of Brute-force Attacks

Brute-force attacks are most frequently detected at the host level by inspecting access logs. If the predefined number of unsuccessful login attempts is reached, an alert is fired, the attacker blocked or other attempts significantly delayed. This approach is effective, even for distributed attacks. The main drawback is that it does not scale well.

We present five detection approaches that profit from the scalability of network flows. The first is a simple analogy of pattern matching known from deep packet inspection. The second approach extends the first one by searching for *similar* traffic instead of fixed patterns. The following two exploit periodicity and the even distribution of attacks in time. And the last one finds abrupt changes in entropy time series.

## 2.1 Signature-based Approach

Similarly to pattern matching in deep packet inspection, signatures can be used in flow-based intrusion detection too. The flow-based signatures describe network traffic by specific values, or ranges of values, of flow features and computed statistics. The signatures are then searched in acquired flows. This is done in separate time windows, typically when exported flows are sent from the collecting process to the metering process. So this simple approach does not consider changes of the monitored traffic in time.

Concerning brute-force attacks, the relevant signature can be comprised of features and statistics describing both requests and replies thanks to the interactive nature of the attacks. The requests carry attempted credentials and the replies information about whether the login was successful or not.

**Method** First, the most popular attacked services such as SSH, Telnet, RDP or web applications using HTTP or HTTPS are run on mostly well-known network ports such as TCP/22, TCP/23, TCP/3389, TCP/80 or TCP/443. Second, the source port of the client (attacker) request, i. e. the destination port of the reply, is usually greater than 1024. Third, login attempts and server replies have a specific (range of) size and duration. These characteristics can be captured by the number of packets and bytes of a flow, its duration or statistics: packets per second, bytes per second and bytes per packet. To sum it up, the signature of an attacker's attempt of SSH authentication may be defined as follows: protocol = TCP, source_port $>$ 1024, destination_port = 22, packets $>$ 10, packets $<$ 30, bytes $>$ 1400, bytes $<$ 5000, duration $<$ 5 s.

Next, selected features of flows matching the given signatures may be analyzed again. For example, in order to determine the number of unique attackers and victims (source and destination IP addresses).

Finally, the number of matching flows is counted for each attacking IP address and if the predefined threshold is reached, an alert is fired. The threshold should express an anomalous number of login attempts in the time window (e. g. 10 in a 5-minute time window for the sample SSH signature above).

**Discussion** The signatures can be implemented as a chain of filters for the nfdump tool [3] or as a decision tree [10].

In 2010, we have deployed a simple signature for SSH attacks in the campus network of Masaryk University to find suspicious hosts conducting SSH attacks. The network consists of about 15 000 networked hosts with public IP addresses including hundreds of SSH servers. The network is open and naturally attracts attackers' attention. The signature itself matched traffic of a few thousand of attackers, but also a few tens of possibly benign hosts from our network. These false positives were caused mainly by hosts connected to a grid and Nagios servers. To eliminate these false positives, we employed the fact that the majority of attack flows is produced by attackers aiming at more hosts within one attack and that the attacks are preceded by scanning the port TCP/22 [9].

In conclusion, the signature-based approach is very straightforward and simple, but for operational use (to eliminate false positives) it is necessary to employ other data sources supporting or contradicting the result.

## 2.2 Similarity-based Approach

Deriving signatures as described above is a time-consuming process. Existing signatures need maintenance as tools and systems generating monitored traffic are evolving and traffic patterns are changing. Additionally, "0-day" attacks are not recognized. We try to address these issues by searching for *similar* flows instead of matching specific signatures. We believe that the similarity of traffic can point to machine-generated traffic, for instance brute-force attacks.

**Method** First, all incoming flows are clustered in a separate time window to isolated groups of similar flows. The similarity is measured by the distance of particular flows (points) in space defined by flow features and statistics. We need to choose a *distance metric function*, its input parameters, i. e. suitable flow features, *radius* used for determination if the flow (point) belongs to groups of flows (points) that are close to each other. For example, we can define points $p_{id}$ representing flows in two-dimensional space as follows: $p_{id} = (pkt, byt)$, where $pkt$ is the number of packets and $byt$ the number of bytes in the flow, and $id$ is a flow identification used in further processing. The distance metric function may be, e. g., the Euclidean metric $d$ given by the Pythagorean formula: $d(p_1, p_2) = \sqrt{(pkt_2 - pkt_1)^2 + (byt_2 - byt_1)^2}$ and the radius a float number.

Second, we assume that flows representing malicious traffic are grouped in clusters[1] and flows representing benign traffic are not similar, therefore they form clusters with a negligible number of members (points). There is also a possibility to discard these clusters for further processing.

Third, IP addresses of flows (points) within each cluster are inspected and the type of the attack is determined. If the cluster contains randomly distributed IP addresses, it may indicate benign traffic. On the contrary, if it is possible to find the same source or destination addresses, it may point out to a multiple or a distributed attack.

Finally, the IP address is classified as the attacker if it generated more than the predefined number of flows.

**Discussion** This is a more generic approach and its detection capability essentially depends on a chosen clustering algorithm and its parameters.

## 2.3 Detection of Automated Actions

Most of the brute-force attacks that we have observed in practice exhibit one similarity that we attribute to their automated behavior: the intensity of an attack from one source *remains relatively constant and periodic* during its course. This attribution is supported by our knowledge of available brute-forcing tools that generally allow their users to set the attack intensity only by specifying the number of attack tasks running in parallel.

Traffic with such property is naturally not unique to brute-force attacks — querying the NTP server, various protocols' keep-alives, IM, etc. behave in a similar way — however, this communication is usually directed to well-known machines or ports that are generally not targets of attacks. These can be thus easily discarded beforehand.

**Time Window Heuristic** Detection of traffic with almost constant intensity can be done using a simple heuristic. Any machine attacking with constant intensity and with zero or fixed delays between attack attempts will create only slightly varying number of flows in any two time windows. This number can be influenced by e.g. network conditions or machine slowdowns.

Figure 1 illustrates two attacks that are both periodic and constant-intensive. There are on average 250 attempts every 6 minutes. The burst attack does not

[1]Some clusters may contain traffic generated by port scanning and some other actual password guessing.

have fixed delays between attacks as it concentrates these attempts into the first 2.5 minutes, whereas the continuous attack distributes the attempts evenly.

The ability to identify this traffic as potentially malicious using the aforementioned heuristic largely depends on two properties of the used time windows. First, there is a size of a time window. As can be clearly seen in Figure 1, short time windows (dark rectangles) may be good enough to detect an attack that does not vary much, but may fail with more complicated patterns. Longer time windows (light rectangles) may detect even the burst attack, but may fall short for attacks that do not last long. Second, there is the relative temporal distance of time windows. Fixed distances can suffer from local non-uniformity of attack intensity when a time window is shorter than the execution of all attack tasks running in parallel. Random distances with potential overlapping seem like a better choice to counter this scenario.

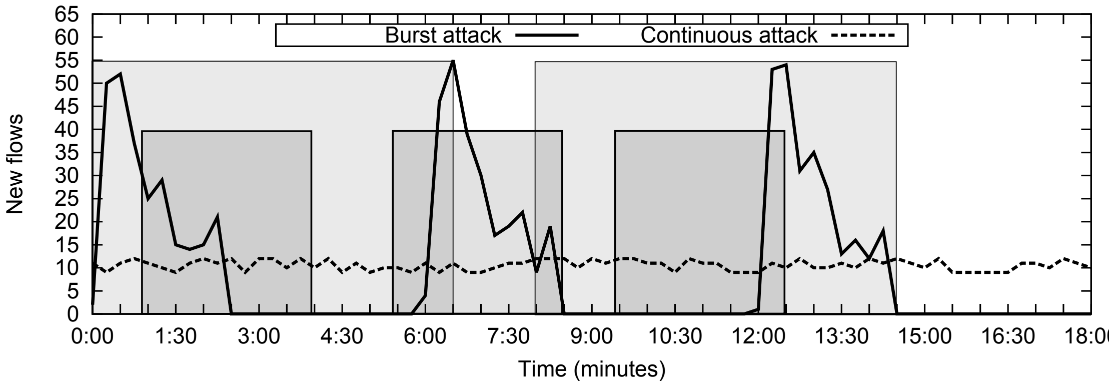

Figure 1: Visualization of two types of attacks and two possible time window settings.

**Fourier Transform** Another approach to identify almost constant periodic traffic is to employ methods of signal processing. The basic premise is that a network traffic trace can be viewed as a signal that can be processed further. This approach was already used in [2], [5] and by others to detect network anomalies. These authors basically used signal processing methods to find deviations in traffic to identify an attack. In this subsection we are focusing more on the problem of finding unwanted regularities in a traffic flow to identify ongoing attacks.

A Fourier transform (FT) basically generates a frequency spectrum of a given signal, i.e. decomposes the signal into a set of simple oscillating functions. The results of a FT show which frequency components are dominant in the signal. Figure 2 shows the result for the burst and the continuous attacks. Peaks in the burst attack transformation point to a presence of a dominant repeating pattern, i. e. the attack. On the other hand, a FT of a continuous attack does not reveal anything and is akin to a FT of non-malicious traffic.

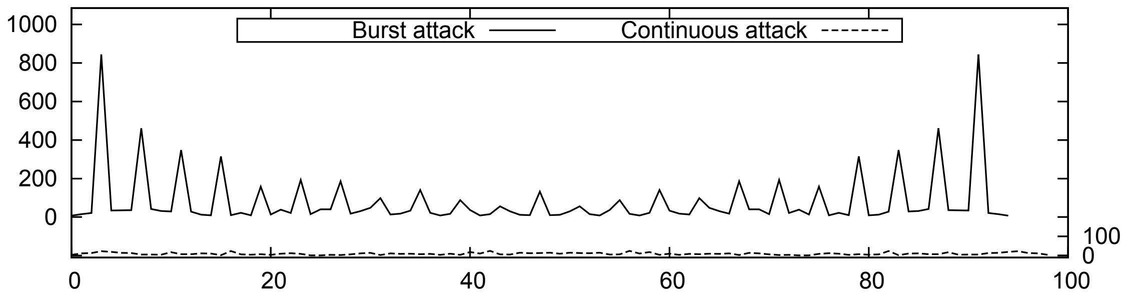

Figure 2: Fourier transform of attacks in Figure 1. *Burst attack is offset 200 points up the y-axis for clarity.*

**Discussion** Both, the window heuristic and the Fourier transform allow the discovery of almost constant periodic traffic that may be a symptom of an ongoing attack. Each method fills a specific niche; the heuristic is more useful for evenly distributed attacks, the Fourier transform for attacks with more complicated patterns.

## 2.4 Detecting Abrupt Changes in Entropy Time Series

In [6], entropy was proposed as a metric to discover a wide range of anomalies in flow traces. Entropy can reduce high-dimensional network traffic to a single numeric value. While this simplification inevitably implies a loss of information, it has the benefit of reduced computational complexity. The use of entropy makes it possible to discover various occurrences of brute-force attacks, such as (D)DoS, large-scale scanning activity and worm outbreaks in the observed flow traces since these types of attack heavily modify the distribution of the underlying flow features.

**Method** In [11], we proposed to use the normalized Shannon entropy to form time series for five flow features: Source and destination IP address, source and destination port and packets per flow. The normalized Shannon entropy normalizes all entropy values to the interval [0, 1] so that all five time series become directly comparable.

Every data point for the resulting five time series is computed by calculating the entropy over a sliding window which consists of all observed flows within the last five minutes. In addition, the sliding windows overlap by four minutes. The overlap should lead to more fine grained results at the cost of being computationally slightly more expensive.

Figure 3 illustrates an example. The x-axis depicts time whereas the y-axis shows the normalized Shannon entropy ranging from 0 to 1. The spike in all time series at May 26 at around 7am could be considered an abrupt change which should be detected by the proposed algorithm.

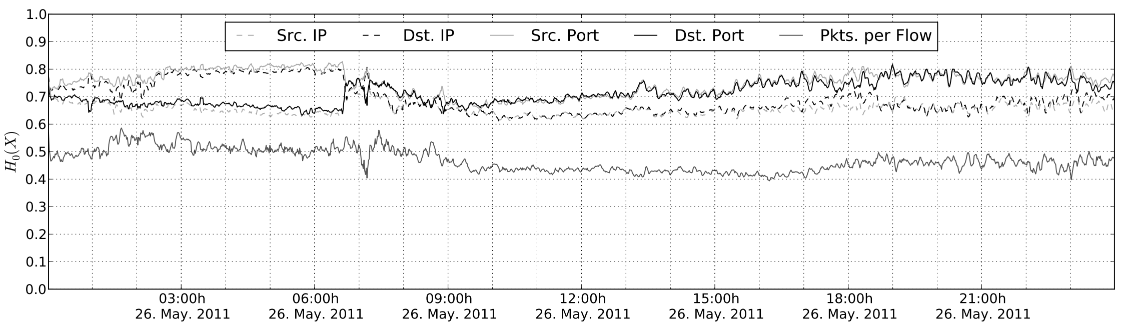

Figure 3: Entropy time series for five flow features.

The main idea of the proposed algorithm is to continuously conduct *short-term predictions* about the progression of the respective time series. The simple exponential smoothing algorithm is used for predictions. After every prediction, the difference to the measured entropy value, the so-called prediction error, is computed. The higher this difference, the more abrupt can the change in the time series be considered.

However, the prediction errors are not equally significant since the underlying entropy time series exhibit different statistical distributions. To make all five time series equally significant, we multiply them with a weight factor which is calculated based on the empirical standard deviation of the respective time series. Finally, all five prediction errors are summed up to a single anomaly score. This makes it possible to configure a single threshold. As soon as the anomaly score exceeds the configured threshold, an alert can be fired.

**Discussion** The evaluation and real-world experiments conducted in [11] suggest that the algorithm scales very well. In addition, the approach is easy and fast to deploy.

# 3 Flow Limitations

The previous sections showed how it is possible to use network flows to discover anomalies and intrusions. Using network flows for attack and anomaly detection also has its downsides which are discussed below.

## 3.1 Information Loss

When network traffic is converted to network flows, certain information remains (IP addresses and ports) and some information is derived (bytes per flow or packets per flow). Other information, like the packet payload, is inevitably lost. This loss of information implies that network flows are not suitable for the detection of all kinds of malicious network activity. This is not an issue in case of flow-based brute-force attack detection, but attacks which manifest solely in packet payload, such as remote exploits, are virtually invisible in network flows.

## 3.2 Collection Delays

Flow-based detection does not happen in real-time because flow exports are generally driven by three timeouts. The *active* timeout splits long-lasting flows and the *passive* timeout exports slow flows. These two timeouts influence flow aggregation considerably and therefore the detection based on volume characteristics. The third timeout is used for periodically sending acquired flows from the metering to the collecting process. Traditionally, the size of this time window is 5 minutes. That means, the acquired flows are sent at least with a 5-minute delay which could be considered too late in case of attack which happen very fast.

## 3.3 Packet Sampling

In [1], Brauckhoff et al. evaluated the influence of packet sampling on anomaly detection. The authors made use of an unsampled data set containing activity of the Blaster worm. Using a traffic baseline which lacks the activity of Blaster, the authors simulated packet sampling at increasing rates and could thus estimate the impact of packet sampling.

Their findings suggest that with increasing sampling rates, the number of bytes or packets is still useful to detect Blaster activity. The number of flows is strongly biased by sampling, though. Besides, the authors conclude that entropy-based flow summarization is less affected by packet sampling than volume-based summarization such as bytes or packets per flow.

# 4 Detection Evasion

The methods proposed in Section 2 operate with several assumptions and constraints that dictate what types of attack each method can detect. If these constraints are taken into account by an attacker, it is possible to come up with relatively straightforward methods of detection evasion that can severely limit the potential of proposed methods.

## 4.1 Attack Function Separation

As was described in [4], the SSH attack typically has three phases: Scanning, Brute-force and Die-off. In our experience, this scheme is not limited to SSH attacks and can be applied to other protocols and authentication schemes. Flow-wise, the scanning phase is identified by large amount of small flows destined to a large number of targets. There are usually lots of unsuccessful connections to closed or filtered ports. In the brute-force phase, recorded flows are larger and destined to specific targets with virtually no unsuccessful connections. The die-off phase that is equal to a successful attack has usually few large and infrequent flows.

Both, [9] and [4] expect to some extent a specific attacking machine to go from Scanning phase over Brute-force phase and eventually to Die-off phase. This is a valid heuristic for attackers with limited number of machines at their disposal. However, attackers with more resources can allocate their machines into groups specialized on each phase, thus evading this heuristic with only a little overhead.

## 4.2 Hiding Under Threshold

Since the flow-based detection methods cannot inspect data in the same detail as deep packet inspection, they are inherently dependent on using thresholds to differentiate between normal and deviant behavior. These thresholds are usually tailored towards a given network and either determined manually or automatically. Looking at some methods in recent papers, e.g., [7], it is apparent that the general approach is to use conservatively high thresholds to avoid false positives. These methods are suited for noisy attacks and can be subverted by limiting the attack intensity. Attackers can thus stretch their attacks in both, time and volume in order to stay under the respective thresholds and evade detection.

## 4.3 Temporal Distortions

Commonly available brute-forcing tools (e.g., AccessDiver, Sentry, Hydra) allow attackers to configure the number of threads that will run in parallel. This effectively determines the average amount of attack attempts per given time window. This amount remains relatively constant during the course of an attack. This behavior is exploited by methods presented in Section 2.3 to detect attacking machines.

By introducing temporal distortions (e. g., random delays) to traffic flows, the attacker can disrupt the periodicity and the automated profile of an attack, thus rendering the detection method ineffective. The NCrack brute-forcer is going partly this way by adapting its brute-forcing process automatically to network conditions (e.g., waiting with the next attack attempts after blocking of a machine or proxy), but to the authors' knowledge, there is no available brute-forcer to employ this relatively cheap and easy technique.

### 4.4 Flow Stretching

Some detection methods that were discussed so far relied on particular flows being relatively short and only a few packets in volume. This is reasonable, given that attacking machines usually exchange the bare minimum of data in order to try to authenticate. Flows of a particular attack thus look alike; the only difference is a few bytes, usually influenced by username and password length. There is, however, danger in such an approach. Several important protocols — SSH, RDP and HTTP(S) — by design allow the exchange of arbitrary data in the process of authentication, before the final decision whether to let an attacker in or not is made. This arbitrary data can be used to inflate both the flow size and the flow duration. Such inflated flows no longer look alike. Instead, they can be made to look like valid communication and avoid detection.

## 5 Summary

This chapter gave an overview about current research in the field of flow-based attack and anomaly detection. We summarized state of the art concepts for attack detection, especially brute-force ones, and concluded the chapter by discussing evasion strategies as well as the limitations inherent to the process of detecting attacks in network flows.

## References

[1] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of Packet Sampling on Anomaly Detection Metrics. In *Internet Measurement Conference*, pages 159–164, Rio de Janeiro, Brazil, 2006. ACM.

[2] Christian Callegari, Michele Pagano, Stefano Giordano, and Teresa Pepe. Combining wavelet analysis and information theory for network anomaly detection. In *International Symposium on Applied Sciences in Biomedical and Communication Technologies*, pages 1–5, Barcelona, Spain, 2011. ACM.

[3] Peter Haag. NFDUMP. `http://nfdump.sourceforge.net/`.

[4] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras. SSHCure: A Flow-Based SSH Intrusion Detection System. In *International Conference on Autonomous Infrastructure, Management, and Security*, Luxembourg, Luxembourg, 2012. Springer.

[5] Chin-Tser Huang, Sachin Thareja, and Yong-June Shin. Wavelet-based Real Time Detection of Network Traffic Anomalies. In *SecureComm and Workshops*, pages 1–7, 2006.

[6] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining Anomalies Using Traffic Feature Distributions. *SIGCOMM Comput. Commun. Rev.*, 35(4):217–228, 2005.

[7] Martin Rehak, Michal Pechoucek, Pavel Celeda, Jiri Novotny, and Pavel Minarik. CAMNEP: agent-based network intrusion detection system. In *International Conference on Autonomous agents and multiagent systems*, pages 133–136, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[8] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An Overview of IP Flow-Based Intrusion Detection. *Communications Surveys Tutorials, IEEE*, 12(3):343 –356, quarter 2010.

[9] Jan Vykopal. A Flow-Level Taxonomy and Prevalence of Brute Force Attacks. In *Advances in Computing and Communications*, pages 666–675, Kochi, India, 2011. Springer.

[10] Jan Vykopal, Tomas Plesnik, and Minarik Pavel. Network-Based Dictionary Attack Detection. In *International Conference on Future Networks*, pages 23–27, Bangkok, Thailand, 2009. IEEE Computer Society.

[11] Philipp Winter, Harald Lampesberger, Markus Zeilinger, and Eckehard Hermann. On Detecting Abrupt Changes in Network Entropy Time Series. In *Communications and Multimedia Security*, pages 194–205, Ghent, Belgium, 2011. Springer.

# Index

attack function separation, 10

brute-force attack, 1

cluster, 4

distance metric, 4

encrypted traffic, 2
entropy, 7

flow stretching, 11
flow-based attack detection, 1
flow-based signatures, 3
fourier transform, 6

hiding under threshold, 10

information loss, 7, 9

noisy attack, 2

packet sampling, 9

signal processing, 6
stealthy attack, 2

temporal distortions, 10
time series, 7
time window heuristic, 5