HOW TO DETECT MIDDLEBOXES: GUIDELINES ON A METHODOLOGY

Vahab Pournaghshband[1], Sepideh Hashemzadeh[2] and Peter Reiher[3]

[1]Computer Science Department, California State University, Northridge, USA
vahab@csun.edu
[2]IEEE Member
hashemzadeh.s.h@ieee.org
[3]Computer Science Department, UCLA, Los Angeles, USA
reiher@cs.ucla.edu

ABSTRACT

Internet middleboxes such as VPNs, firewalls, and proxies can significantly change handling of traffic streams. They play an increasingly important role in various types of IP networks. If end hosts can detect them, these hosts can make beneficial, and in some cases, crucial improvements in security and performance But because middleboxes have widely varying behavior and effects on the traffic they handle, no single technique has been discovered that can detect all of them.

Devising a detection mechanism to detect any particular type of middlebox interference involves many design decisions and has numerous dimensions. One approach to assist with the complexity of this process is to provide a set of systematic guidelines. This paper is the first attempt to introduce a set of general guidelines (as well as the rationale behind them) to assist researchers with devising methodologies for end-hosts to detect middleboxes by the end-hosts.

The guidelines presented here take some inspiration from the previous work of other researchers using various and often ad hoc approaches. These guidelines, however, are mainly based on our own experience with research on the detection of middleboxes. To assist researchers in using these guidelines, we also provide an example of how to bring them into play for detection of network compression.

KEYWORDS

Detection, Middlebox, Guidelines

1. INTRODUCTION

Abstractly, we often assume that the Internet follows the end-to-end principle, with smart endpoints and a dumb network. However, this general picture is very different from the capabilities of the latest technologies and the actual Internet is far more complex, with the emergence and rapidly growing prevalence of middleboxes deployed at various points in the network.

Middleboxes are defined as intermediary devices which take actions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host [1]. They manipulate traffic for purposes other than simple packet forwarding. In addition to routing the traffic, middleboxes can make serious changes to network flows from altering the user-data to more transparent effects such as imposing additional delay on the traffic. These influences by third party middleboxes could be for malicious, security, or performance reasons.

A wide variety of middleboxes have been proposed, implemented, and deployed during the last decade [2,27,28]. Today's enterprise networks rely on a wide spectrum of specialized applications of middleboxes. Middleboxes come in many forms such as proxies, firewalls, IDS, WAN optimizers, NATs, and application gateways, and are used for various purposes including performance and security improvement and compliance. They are an integral part of today's Internet and play an important role in providing high levels of service for many applications. Recent papers have shed light on the deployment of these middleboxes [2, 3] to show their prevalence. And a recent study [4] shows that the number of different middleboxes in an enterprise network often exceeds the number of routers. Trends such as proliferation of smartphones and wireless video are set to further expand the range of middlebox applications [5].

In some cases, middleboxes do indeed exert a real influence on traffic. In others, they merely act as if they are simple routers. Knowing the presence of middleboxes is most critical in the former case, when they are actually doing something to the traffic. This is also the easier case to detect, since a middlebox that does nothing leaves no traces of its presence. We concentrate on this more important case.

Knowing the existence of the influence of middleboxes could be beneficial to the end-hosts. Sometimes the end-hosts would behave differently based on what they sense is happening to their traffic. In such cases, an accurate detection of what is happening to that traffic is the first step. Here, to illustrate this idea, we present a number of scenarios from different categories.

Scenario I

Assume a sender is about to send sensitive data, making encryption necessary. In this case, the sender will check to determine if a strong end-to-end encryption (VPN) on the path is deployed. If he detects that strong encryption is already in place, to save energy and resources, he might choose not to encrypt the traffic stream, since encryption is a relatively expensive operation.

Scenario II

The sender detects that the receiver is using a wireless connection, but is unsure if that connection is secure. If he detects that the last link is unencrypted, he would either refrain from sending sensitive information or would apply end-to-end encryption to the channel. For example, Amazon does not provide end-to-end SSL encryption to its users who are not logged in and does not require them to log in until they are about to make the payment. This is perhaps due to lack of available resources required to encrypt all users' contents for all users. Amazon servers, to use their resources effectively while protecting users' privacy from profiling, could first sense whether the user is using a secure wireless connection or not, and then apply end-to-end encryption only if the user needs that protection. Conversely, the receiver would mark the incoming data as untrusted if he detects that the sender's wireless link is insecure.

Scenario III

An Internet user in an oppressive country might detect Internet censorship imposed by his ISP and then chooses to use a proxy to bypass it. In a different scenario, an Internet user detects wiretapping on his network and uses evasive techniques such as Tor or a VPN.
Devising a detection mechanism for middleboxes can be difficult. For instance, detecting a third party middlebox that does not alter the user data, from the end hosts' point of view, is particularly

challenging, since the effect of middleboxes of this class seems transparent to the end hosts. This is true if either the third party undoes the changes made to the data before passing it to the receiver (e.g., VPN gateways or link-layer compression) or it does not alter the data at all and affects the traffic stream similarly to normal network variation (e.g., the delay-attack [6] by a compromised node in sensor networks or the Shrew attack [7] that selectively drops packets).

There have been numerous efforts to detect various types of middleboxes in the past [8, 9, 10, 11]. However, the proposed approaches are ad hoc and mostly designed to detect only specific types of middleboxes. Despite these differences, nevertheless, there are common elements in the process leading to the design of such methods. These common elements could be identified and summarized into general guidelines.

The ultimate objective of this paper is to assist researchers who intend to conduct research focused on detection of the interference of middleboxes and introduce them to the potential challenges they might face in the process. In addition, we present some recommendations on how to overcome those challenges in certain situations. In other words, the desirable outcome we seek here is to assist with devising an accurate detection mechanism in an efficient manner with the help of systematic guidelines that have been drawn from past experiences. To the best of our knowledge this is the first attempt to devise systematic guidelines for the purpose of assisting other researchers.

While introducing the phases and steps of our proposed guidelines, we demonstrate each by applying it to an example: end-to-end detection of network compression on the path. The network compression detection approach and the corresponding results have been presented in [12] as part of our prior work on this subject.

Detecting Network Compression: A Case Study

One way to increase network throughput is to compress data that is being transmitted. Network compression may happen at different network layers and in different forms: application layer, TCP/IP header [13], IP payload [14], and link layer [15, 16].

Except for application-layer compression, compression happens at intermediate nodes, often without the knowledge of end-users. For example, in January 2013, a researcher discovered that Nokia had been applying compression to its users' data without their knowledge [16]. In this case, the intermediary was surreptitiously decrypting and re-encrypting user packets in order to effectively apply compression. Users surely would have preferred to know that this was happening, both because of the security risk and because it would render their own application-level compression unnecessary.

However, performing compression and decompression requires many resources at the intermediate nodes, and the resulting overhead can overload the intermediary’s queue, causing delay and packet losses. Further, not all commercial routers come with compression capabilities [17]. Thus, some intermediaries apply compression, some do not, and generally they do not tell end-users whether they do. While managing resources effectively at end-hosts is not as crucial as it is at routers, it is still beneficial—particularly for mobile devices where resources are limited. Wasting these resources on redundant compression is undesirable. End-hosts can benefit from recognizing when compression has already being applied on a network connection.
Ideally, end-hosts and intermediaries should coordinate their compression decision, but practical problems make that ideal unlikely. Therefore, since the end-hosts have the greatest interest in proper compression choices for their data, they could detect if intermediate compression is present and adjust their behavior accordingly. An end-to-end approach to detect compression by

intermediaries can help to save end-host's resources by not compressing data when intermediaries are already doing so.

The remainder of the paper is organized as follows: Section 2 presents the guidelines for a detection methodology, followed by related work in Section 3. Section 4 concludes the paper.

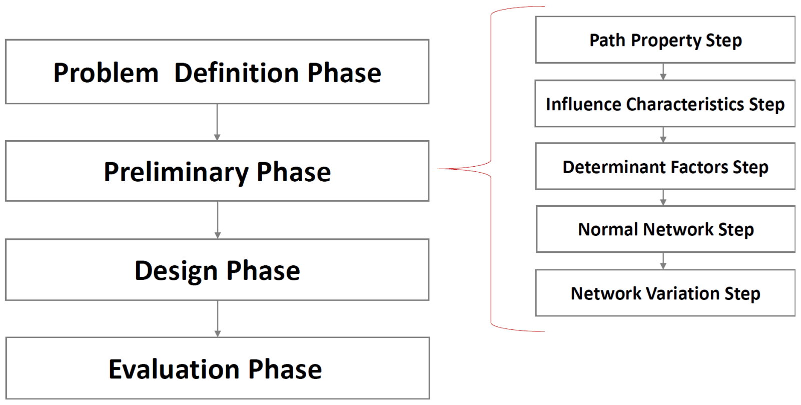


Figure 1: Phases and steps in the guideline.

2. GUIDELINES ON A DETECTION METHODOLOGY

In this section; we present the guidelines for a detection methodology. Later in this section, we show that a careful completion of the preliminary phase will lead to a significant reduction of the complexity in the design phase.

Our proposed guidelines consists of phases in a sequential order. Figure 1 illustrates the overview of the phases. The phases introduced in this guideline are:

- Phase 1: Problem Definition Phase
- Phase 2: Preliminary Phase
- Phase 3: Design Phase
- Phase 4: Evaluation Phase

In the problem definition phase we clearly define the problem and its scope by setting the assumptions. The preliminary phase consists of five steps to be followed in the presented order. The design phase in this process is when, with the help of information obtained in the preliminary phase, the detection algorithm is developed. The evaluation takes place when we validate the proposed detection mechanism devised in the design phase.

2.1 Problem Definition Phase

2.1.1 What is it that we want to detect?

What it is that we want to detect should be clearly stated. For instance, the running example we use in this paper is that we want to detect the presence of network compression on the path between two end-hosts.

2.1.2 Assumptions

The scope of the general detection problem is very broad. Many variations of the problem can be formulated based on constraints and limitations imposed on the problem: e.g., the method used for detection and the degree of detection for the problem. Before we begin with the design of any detection mechanism, we should prepare a list of assumptions we make in order to define the scope of the problem we aim to solve.

2.1.2.1 Degree of detection

A problem can be formulated to only answer an existential question about whether the network flow is influenced by some third-party or not. An instance of this problem can attempt to further characterize the influence to discover exactly what the third party is doing to the network flow.

2.1.2.2 Locating the third-party node/link

The problem statement can be phrased in several ways from determining whether the path is influenced or not to precisely locating the third-party on the path or the link(s) influenced by it.

2.1.2.3 Method of detection

There are essentially two methods of network measurements for detection available to the end-hosts: active and passive measurements. Passive measurements have the goal of minimally affecting the measured network, by merely monitoring traffic on the network and inferring measurements from the observed traffic. Active measurements involve interacting with the network to make measurements, usually by sending probe packets. In active measurements, we refer to the two end-hosts as the sender and the receiver. Based on how the receiver cooperates (if it does) in the detection process, we present three variations to this problem:

A. **Receiver is uncooperative**
   The receiver does not respond to the sender's requests that are beyond their primary purpose of communication. For examples, most web servers expect only HTTP requests and responses.

B. **Receiver is responsive**
   The receiver responds to the sender's requests beyond their primary purpose of communication as long as it does not require any changes on the receiver's machine. For example, the receiver responds to the sender's ICMP requests.

C. **Receiver is cooperative**
   The receiver is willing to make necessary changes on its machine or system to fully cooperate with the sender in the detection process.

2.1.2.4 Assistance from intermediaries or other parties

If all or some intermediaries on the path are responsive, active measures can also be used to get intermediaries to respond with valuable information. In another instance of the problem, the end-hosts assume that the intermediaries are uncooperative. Another instance of the problem is where the end-hosts make use of help from volunteer nodes on the network that are not on the path.

2.1.2.5 Detection by comparing to unperturbed channel

One instance of the problem is when end-hosts have a model of the channel in the absence of a third-party's influence (perhaps captured in the past). In this case, the presence of the third party could be determined by comparing the current network behavior to the model. The other instance of this problem, however, is when the end-hosts do not have access to such information.

2.1.2.6 Static vs. dynamic third-party middleboxes

The last instance we address here is whether the middlebox is static or dynamic. In other words does its behavior remains unchanged at the time that an end-host is trying to detect it or does the third party middlebox notices the end-host’s detection attempt, and hence, evades detection by changing its behavior during the detection period. In the latter case, only a stealth detection mechanism might be effective.

For our running example, end-to-end detection of network compression, we choose the following assumptions, to define the scope of the problem:

Regarding the degree of detection, we only seek to detect the presence of the intentional influence on the network flow. This problem is not about specifically locating the third-party or the influenced link on the path connecting the end-hosts. We assume end-hosts are fully cooperative and can use active measures for detection. However, due to the end-to-end nature of our problem definition, we exclude the scenario where the intermediaries are cooperative. For instance, the end-hosts should not rely on responses from pings sent to the intermediate routers. For the same reason, we exclude the instance where the end-hosts use help from volunteer nodes on the network that are not on the path. Furthermore, our proposed problem assumes that the end-hosts do not have any model of the unperturbed channel (i.e., in absence of the third party) between the end-hosts. We also assume that the end-hosts are not just normal network users and will use their resources to any required level (with some limitation on the available resources) in the detection process. And lastly, we base our detection methodology on detecting only static middleboxes.

2.2 Preliminary Phase

We believe that when starting on the design of a detection mechanism, it is necessary to carefully find answers to the following questions before offering any detection mechanism. In this phase, answering the five questions stated below will defines the steps to complete this phase.

In every step of this phase, we ensure that in addition to clearly addressing the question, and the scope of it, we also address the following questions about it: (1) Why is having an answer for this particular question important, or at least useful in the detection process? (2) What are the potential challenges in the process of answering it?

Some of the steps in this phase require a thorough understanding of the current state of technology and available tools; some others require careful analysis and examination, and some require extensive experimentation. Here, we leave the technical details out of these steps and limit ourselves to the high level presentation of the sub-problem and the expected outcome. Clearly, if a step requires experimentation that means that even more questions must be answered−such as how exactly to implement it, what would be a suitable environment to run it on, or how to prepare the experiment environment.

1) *The Path Property Step "P":* What important pieces of information about the network path properties are available to typical end-hosts?

Similar to any detection process in the real world, special tools are needed to observe and to look for signs leading to detection. Hence, it is crucial to know what information and tools are available that are observable and measurable, by the end-hosts, in order to utilize them effectively for the detection process.

For instance, some network properties such as RTT, packet delay variation, available bandwidth, and hop count (leaving out the discussion on the measurement accuracy) can be measured by the end-hosts. On the other hand, other invaluable information such as the queue size of the routers on the path is normally not available to typical end-hosts. Technically, we are only interested in standard methods and tools available for standard computers with standard equipment. For instance, we assume that in an end-host, processing time and packet arrival time can be measured and its TCP congestion control mechanism's behavior is observable. On the other hand, we exclude using irregular methods to obtain some information. For instance, we do not consider using measuring electromagnetic emissions to detect the queue size of a router.

Throughout this paper, we refer to the set of all available network properties about the path as *P*.

Challenges:

- Understanding exactly how well the properties in *P* can be measured and how the existing tools' imprecision and potential inaccuracy in measurements would affect the accuracy of the detection mechanism.

Detecting Network Compression: The receiver can measure the arrival time of packets with precision on the order of at least micro-seconds.

2) *The Influence Characteristics Step "I":* What are the characteristics of the influence *I*?

To detect the presence of *I*, we must have a way to distinguish *I* from other types of influences. Therefore, it is necessary to find unique, indicative, and distinguishing characteristics of *I*.
This step requires gathering and understanding all, if any, publicly available and known information about the internal design of the influence in question. A thorough and perhaps creative analysis is necessary to find subtle unique characteristics about *I* that can be used to detect it. These characteristics of *I* will then be exploited to help detect the middlebox. Technically, we are mainly interested in observable and measurable effects on elements of *P* and not interested in hidden or non-measurable effects of *I*. For instance, one characteristic of gateway VPNs is the relatively constant delay they impose on every packet due to the encryption/decryption time.

Challenges:

- These characteristics are not always obvious and might require a thorough and careful examination to find them.

Detecting Network Compression: If compression is placed on the bottleneck of the path, then end-hosts would potentially sense a higher bandwidth when data with lower entropy is sent. In practice, this is the primary, if not the only, objective of employing network compression.

3) ***The Determinant Factors Step "D":*** **What elements in *P* are impacted by *I* (and to what extent)?**

The values of *P*, by definition, are all that the end-hosts can know. Therefore, the only way to detect *I*, is by examining the path properties, and by looking at the changes in their values in the presence and absence of *I*.

This is an important step in identifying the factors involved in detecting *I*. We define *D* as the set of all indicative elements of *P* in detecting *I* ($D \subseteq P$). Intuitively, if the value of a path property remains constant in the presence or absence of *I*, then it is not a helpful piece of information in the detection process. It is also important to determine which of these elements are more indicative than the others in detecting *I*. Furthermore, any interdependency relationship between members of *D* must be investigated.

Theoretical analysis could lead to a hypothesis on *D*, where further experiments could verify it. Another approach to this step could be the use of network simulations. One could simulate the effects of *I* in a clean environment (i.e., in the absence of network variation and any types of traffic other than the one generated by the detection probes) and look for changes in values of various path properties in *P*.

Challenges:

- How exactly to assess the relative level of importance of $d \in D$ in detecting *I*?
- How do inaccuracies in measurements influence our findings?
- How exactly to derive the interdependency relationship between members of *D*?

Detecting Network Compression: Network compression is designed to improve, and hence should affect, the available bandwidth. Therefore, the available bandwidth is a strong candidate as a determinant factor that is potentially influenced by network compression.

4) ***The Normal Network Step "N":*** **What are the sufficient, yet unavoidable, assumptions about the normal network behavior, in the particular network environment, required to make any comments on detectability of *I* feasible?**

Even if we make the assumption that the end-hosts have no access to information on the unperturbed channel, for any detection mechanism to work, there should be a precise definition for the notion of normal behavior to use as a reference point. This is where we define the specific network environment (either in high level properties or low level). For instance, if it is multi-hop wireless network, then we expect a higher rate packet loss, whereas in a wired network, random losses are rare events. Another approach is to impose an upper-bound on the value of RTT.

These are conditions, assumptions, and constraints on the elements of *D*. But only those that are in *D* are significant, because *P* - *D*, by definition, is expected to remain relatively constant. Hence, there is no need to check whether they have deviated from normal behavior.

Basically, the violation of assumptions on the elements of *D* would be used to indicate the existence of deviations from normal network behavior.

Challenges:

- What is precisely "normal" and how do we define it?

- It is crucial to come up with not only correct, but tight assumptions about normal network behavior. Failing to arrive at correct assumptions would lead to false positives. On the other hand, loose assumptions almost certainly lead to undesirable false negatives.

Detecting Network Compression: In a normal network (i.e., in the absence of compression on the path), all packets of the same size are treated equally regardless of the data entropy of the content of their payloads.

5) *The Network Variation Step "V":* Does normal network variations (e.g., network congestion, load balancing effects, link failures, and dynamic routing effects) influence the end-to-end detection of *I*?

Any proposed detection mechanism should work in a real network. An approach that only works in a controlled and isolated environment is not very useful.

If the answer is "no" to the question posed in this step, then we can completely skip this step. For instance, detecting a third party that sends out spoofed control packets is not affected by normal network variation.

However, if the answer is "yes" (which is the case for all delay and loss-based influences), then we proceed to the following questions.

- What is the specific set of normal network variation that we care about in the detection process, *V*? For instance, one could focus only on network congestion due to its popularity.
- How well is *I* distinguishable from effects of *V*?

Note that it is generally true that congestion usually hinders the detectability of loss or delay-based influences. Clearly, if congestion, in some cases, helps the detection process, one could impose network congestion to makes detectability easier.

Challenges:

- Network variations that resemble normal loss and delay in the network are particularly challenging. This is because intentional and normal loss or delay in those cases are perhaps not easily distinguishable. This leads to another issue: how is *I* distinguishable from *V*?
- Active probing used for the detection process may contribute to network congestion.

Detecting Network Compression: Congestion influences the available bandwidth.

2.3 Design Phase

This is a crucial phase in the process where we use the information obtained in the preliminary phase to produce the detection algorithm. The design phase requires creativity and knowledge of the subject to use the information gathered in the previous phases. We recommend testing the initial approach in a simulated environment to verify the approach. Then information from Step *V* could be incorporated to produce an approach that works well in real networks. Again, we emphasize that the goal is to detect the middlebox's interference or influence on the traffic flow and if it is not interfering with the network, whether it is detectable or not is not an objective of this paper. Also, note that $\forall d \in D$ *is* used in the detection mechanism, and inaccuracies involved in measuring each element *d* must be taken into consideration.

Using the example of detecting network compression we show how the information obtained from the previous phases could lead to the design of an approach that to detects this particular influence.

Detecting Network Compression: To detect if compression is provided on the network we exploit the unique effects of compression on network flows. Assuming the original packets are of the same size, compressed low entropy data packets are expected to be considerably smaller than compressed packets containing high entropy data (the sensed bandwidth is vastly different), which in turn leads to a shorter transmission delay. Based on these facts, the sketch of our approach is as follows:

Send a train of fixed-size packets back-to-back with payloads consisting of only low entropy data. Then send a similar train of packets, except these payloads contain high entropy data instead. The receiver then measures the arrival times of the first and the last packet in the train, independently for low entropy (t_{L_1} and t_{L_N} , where N is the number of packets in a single train) and high entropy (t_{H_1} and t_{H_N}) packet trains. Note from step *P* of the preliminary phase that the typical machines are capable of measuring the arrival time of packets with a precision on the order of at least micro-seconds. Since the number of packets in the two trains is known, and all of the packets have the same uncompressed size, the following inequality will hold if some kind of a network compression is performed on the path:

$$\Delta t_L = t_{L_N} - t_{L_1} < \Delta t_H = t_{H_N} - t_{H_1}$$

This inequality suggests that the total set of highly compressible low entropy packets gets to the destination faster than the set of less compressible high entropy packets since the available bandwidth is better utilized by compression. Conversely, if the packets are not being compressed by any intermediary, then the two sides of the inequality should be almost equal. This suggests that a threshold should be specified to distinguish effects of compression from normal Internet variabilities:

$$\Delta t_H - \Delta t_L > \tau$$

The underlying rationale behind this approach is that because of the presence of network compression on the bottleneck link, the receiving party should sense a relatively higher bandwidth when the train of low entropy data is sent. This is because the same amount of data is received in both cases, but in a significantly shorter period of time when low entropy data is used.

2.4 Evaluation Phase:

Any proposed detection mechanism must be not only validated, but also evaluated under certain or all network conditions to confirm that it successfully detects the influence in question. This requires answers for questions such as:

1) How can we validate/evaluate our proposed detection algorithm?
2) How confident are we in the results of the evaluation system?
3) What metrics should we use to evaluate our detection mechanism?
4) Does the testbed used to validate our proposed detection mechanism have any shortcomings that would impact the accuracy of evaluation of our detection mechanism?

As an example, let's examine perhaps the most popular testbed used in the research community for Internet measurements: PlanetLab [18]. PlanetLab is widely used by the research community and is generally a suitable candidate for such purposes. For every researcher working on PlanetLab there is a need to know what bias the system introduces in the collected experiment results. For instance, one should keep in mind that PlanetLab is not completely representative of the current Internet. This testbed is all about small, long running services in specific locations. There are also some shortcomings associated in the use of PlanetLab as well. For instance, we normally have very little, if any at all, control over the actual path between the chosen sender and receiver. Depending on the nature of the experiment, this could make PlanetLab somewhat ineffective.

Challenges:

- Find a platform to test the proposed detection mechanism on a global testbed, specifically one that allows us to possibly overload its nodes, if needed. It is highly desirable that the chosen testbed gives us some control over or information about the path between the nodes. Known as the ground truth problem, this is a fundamental challenge in validating detection mechanisms and the majority of previous research has acknowledged this problem to some extent. For instance, how are we going to evaluate our compression detection mechanism if we do not know whether link compression exists on the path between two nodes of the testbed?
- What metrics should be used in the evaluation process and how do we accurately measure them: e.g., false positive rate, detection rate, performance overhead, and packet delivery rate?
- Is there a need for a general evaluation system that applies to detecting all influences? How feasible is this idea?
- Recent studies, suggests that testbed results for Internet systems do not always extend to the targeted deployment. For example, Ledie et al. [19] and Agarwal et al. [20] show that network positioning systems perform much worse “in the wild” than in PlanetLab deployment. Identifying such inconsistencies to avoid false claims is not trivial.

3. RELATED WORK

While our work is the first attempt to introduce general guidelines for devising detecting methodologies for middleboxes, there has been much work in the past that proposed various approaches to detect middleboxes’ interference on traffic flows. In this section, we briefly present some of that work that has mainly security applications or implications.

Very few and even more specific works focused on detecting intentional delaying and dropping of selective packets for malicious reasons. Song et al. [5] propose two different approaches to detect and further accommodate delay attacks in wireless sensor networks. Kuzmanovic et al. [21] identify TCP victims (as the first step to detect the attacker) by monitoring their drop rates to help mitigate low-rate TCP-targeted attacks.

Other approaches introduced methodologies to detect middleboxes that send spoofed control packets on behalf of the other party. BTTest [22] focuses on detecting BitTorrent traffic blocking by ISPs that use forged TCP RST packets. Weaver et al. [23] also introduce a method to detect connection disruptions via forged TCP RST packets, but they focused only on detecting the method that China's Great Firewall uses.

FireCracker [24] proposes a framework that, through tailored probes, could be used to blindly discover a firewall policy remotely as a blackbox and without prior knowledge about the network configuration. In a more recent work, SymNet [25] proposes a static analysis technique that can model stateful middleboxes such as stateful firewalls and IDSs. There basis also been work on detecting middleboxes that modify TCP Fields. Tracebox [26] detects middleboxes that modify TCP sequence and acknowledgement numbers, as well as TCP MSS option. Glasnost [11], on the other hand, detects modified TCP advertised window size. Detecting traffic discrimination has drawn more research attention than detecting other types of middleboxes. The majority of such detection mechanisms [8, 9, 11] use the relative discrimination technique, where they test each application (the measured flow) against a flow that is assumed to be non-discriminated (the baseline flow).

4. CONCLUDING REMARKS

In this paper we present a set of general guidelines to assist researchers with devising end-to-end detection methodologies for detecting middleboxes. More detailed, low-level guidelines could provide support for more specific design decisions. While more detailed guidelines arise left for future work, we still need to be careful not to lose the generality of the guidelines here. For instance, the evaluation phase might be generalized into a general evaluation methodology that can be used to validate all methodologies for detecting middleboxes.

We have presented a detailed case study using our proposed guidelines for detecting network compression. Evaluating the outcome of this work is difficult because it is rather a qualitative assessment. While the results are encouraging, more case studies will help to assess the effectiveness of this guideline.

REFERENCES

[1] Brian Carpenter and Scott Brim. “Middleboxes: Taxonomy and issues,” Technical report, RFC 3234, February, 2002.

[2] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. “Making middleboxes someone else's problem: network processing as a cloud service,” ACM SIGCOMM Computer Communication Review, 42(4):13-24, 2012.

[3] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, Zhuoqing Mao, and Ming Zhang. “An untold story of middleboxes in cellular networks,” ACM SIGCOMM Computer Communication Review, 41(4):374-385, 2011.

[4] Justine Sherry, Sylvia Ratnasamy, and Justine Sherry at “A Survey of Enterprise Middlebox Deployments,” 2012.

[5] “Enterprise Network and Data Security Spending Shows Remarkable Resilience,” http://www.reuters.com/article/2011/01/10/idUS166224+10-Jan-2011+BW20110110,January 10, 2011.

[6] Hui Song; Sencun Zhu; Guohong Cao, "Attack-resilient time synchronization for wireless sensor networks," Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on, vol., no., pp.8 pp., 772, 7-7 Nov. 2005

[7] Aleksandar Kuzmanovic and Edward W. Knightly. “Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants,” In Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communications, pp. 75-86, New York, NY, USA, 2003.

[8] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. “Detecting Network Neutrality Violations with Causal Inference,” In Proc. of the CoNEXT Conference, 2009.

[9] Kanuparthy, P.; Dovrolis, C., “DiffProbe: Detecting ISP Service Discrimination,” Proceedings of IEEE INFOCOM, 2010, vol., no., pp.1,9, 14-19 March 2010

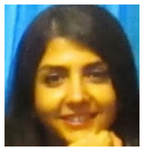
[10] Partha Kanuparthy and Constantine Dovrolis. “ShaperProbe: end-to-end detection of ISP traffic shaping using active methods,” In Proceedings of the 2011 ACM SIGCOMM conference on Internet Measurement Conference, pp. 473-482. ACM, 2011.

[11] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, P Krishna Gummadi,Ratul Mahajan, and Stefan Saroiu. “Glasnost: Enabling End Users to Detect Traffic Differentiation,” In NSDI, pp. 405-418, 2010.

[12] Pournaghshband, V.; Afanasyev, A; Reiher, P., "End-to-end detection of compression of traffic flows by intermediaries," In Proceedings of IEEE Network Operations and Management Symposium (NOMS), 2014 IEEE , vol., no., pp.1,8, 5-9 May 2014.
[13] L. Jonsson, G. Pelletier, and K. Sandlund, "RFC 4995: The Robust Header Compression (ROHC) Framework," Network Working Group, pp. 1–40, 2007.
[14] A. Shacham, B. Monsour, R. Pereira, and M. Thomas, "IP Payload Compression Protocol (IPComp)," RFC 3173, 2001.
[15] R. Friend and W. Simpson, "RFC1974: PPP Stac LZS Compression Protocol," 1996.
[16] "Nokia hijacks mobile browser traffic, decrypts HTTPS data," http:www.zdnet.comnokia hijacks-mobile-browser-traffic-decryptshttps- data-7000009655, 2013.
[17] HP Support FAQs, Using Compression with HP Router Products. http://www.hp.com/rnd/support/manuals/pdf/comp.pdf.
[18] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. "PlanetLab: an overlay testbed for broad-coverage services," SIGCOMM Computer. Communication. Rev, 33(3):3-12, July 2003.
[19] J. Ledlie, P. Gardner, and M. Seltzer. "Network coordinates in the wild," In Proc. of NSDI, volume 7, pp. 299-311, 2007.
[20] S. Agarwal and J.R. Lorch. "Matchmaking for online games and other latency-sensitive P2P systems," In ACM SIGCOMM Computer Communication Review, volume 39, pp. 315-326. ACM, 2009.
[21] Chia-Wei Chang; Seungjoon Lee; Bill Lin; Wang, J., "The Taming of the Shrew: Mitigating Low-Rate TCP-Targeted Attack," Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on, vol., no., pp.137,144, 22-26 June 2009.
[22] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. 2008. "Detecting BitTorrent blocking," In Proceedings of the 8th ACM SIGCOMM conference on Internet measurement (IMC '08). ACM, New York, NY, USA, 3-8.
[23] NicholasWeaver, Robin Sommer, and Vern Paxson. "Detecting Forged TCP Reset Packets," In NDSS, 2009.
[24] Taghrid Samak, Adel El-Atawy, and Ehab Al-Shaer. Firecracker: "A framework for inferring firewall policies using smart probing," In Network Protocols. ICNP 2007. IEEE International Conference on, pp. 294-303. 2007.
[25] Radu Stoenescu, Matei Popovici, Lorina Negreanu, and Costin Raiciu. SymNet: "static checking for stateful networks," In Proceedings of the 2013 workshop on Hot topics in middleboxes and network function virtualization, pp. 31-36., 2013.
[26] Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoit Donnet. "Revealing Middlebox Interference with Tracebox," In Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, pp. 1-8, 2013.
[27] Vahab Pournaghshband, Leonard Kleinrock, Peter Reiher, and Alexander. Afanasyev, "Controlling applications by managing network characteristics," In IEEE International Conference on Communications (ICC), 2012.
[28] Vahab Pournaghshband, Majid Sarrafzadeh, and Peter Reiher. "Securing legacy mobile medical devices," In Proc. of Int. Conf. Wireless Mobile Communication and Healthcare, 2012.

AUTHORS

Vahab Pournaghshband is an Assistant Professor in the Computer Science department at California State University, Northridge. He received his Ph.D. from the Computer Science department at University of California, Los Angeles (UCLA) in 2014. He received his M.Sc. in Computer Science from University of California, Berkeley. Also from UC Berkeley, he received his B.Sc. in Electrical Engineering and Computer Science. Dr. Pournaghshband has done research in the fields of computer networks, computer security, and computing education.



Sepideh Hashemzadeh received her B.S. in Engineering Science from University of Tehran in 2013. She has done research in the fields of embedded systems and computer networks. Her research interests focus on security aspect of embedded systems and computer networks.

Peter Reiher received his B.S. in Electrical Engineering and Computer Science from the University of Notre Dame in 1979. He received his M.S. and Ph.D. in Computer Science from UCLA in 1984 and 1987, respectively. He has done research in the fields of distributed operating systems, network and distributed systems security, file systems, ubiquitous computing, mobile computing, and optimistic parallel discrete event simulation. Dr. Reiher is an Adjunct Professor in the Computer Science Department at UCLA.