Protocol-Independent Detection of Dictionary Attacks

Martin Drašar

Masaryk University
Institute of Computer Science
Botanická 68a, Brno, Czech Republic
drasar@ics.muni.cz

Abstract. Data throughput of current high-speed networks makes it prohibitively expensive to detect attacks using conventional means of deep packet inspection. The network behavior analysis seemed to be a solution, but it lacks in several aspects. The academic research focuses on sophisticated and advanced detection schemes that are, however, often problematic to deploy into the production. In this paper we try different approach and take inspiration from industry practice of using relatively simple but effective solutions. We introduce a model of malicious traffic based on practical experience that can be used to create simple and effective detection methods. This model was used to develop a successful proof-of-concept method for protocol-independent detection of dictionary attacks that is validated with empirical data in this paper.

Keywords: traffic classes, anomaly detection, network behavior analysis

1 Introduction

Conventional methods of deep packet inspection (DPI) are being replaced with methods of network behavior analysis (NBA). The ordinary traffic volume often exceeds units and tens of gigabits per second. This is a problem for classic methods of DPI and signature matching that start to demand more processing power than the current hardware can supply. NBA methods are seen as a potential remedy, because of their ability to work with aggregated data and to detect unknown threats [8].

The development of NBA methods diverged into two branches that pursue different goals using different means. On the one hand, there is the academic research with a lot of theoretical work and application of sophisticated concepts. On the other hand, there is the industry practice of using relatively simple, but effective methods to protect from current threats.

In this paper we try to bring these two branches together by introducing a model of an attack traffic based on practical experiences with network protection in the University's CSIRT[1]. This model can be used to design anomaly detection

[1] Computer Security Incident Response Team – organizational unit responsible for maintaining security of the University's network

schemes that are effective and relatively easy to implement as is demonstrated by creating a proof-of-concept method for detection of dictionary attacks.

This paper is divided into six sections. Section 2 describes dictionary attacks. It also presents current advancements in dictionary attack detection and evaluates their detection schemes. Section 3 introduces the concept of traffic classes to describe attacking traffic. Section 4 is focused on a connection between traffic classes and different types of attacks. Section 5 presents the detection method based on properties of one traffic class and also evaluates its effectiveness. Section 6 concludes the paper.

2 Dictionary Attacks

Dictionary attacks exploit valid authentication mechanism, abusing the fact that users tend to choose weak credentials [6]. It is complicated to detect such attacks, especially low-profile ones, because an isolated attack attempt differs from a legitimate one only by intention and not any measurable properties. That is why they are so prevalent [3], [7], [9]. However, they do not receive enough academic attention as most research is focused on DoS attacks, botnet activity and other disturbant behavior [8].

There is a large variety of industry-provided tools to detect dictionary attacks. They are often tied to one particular service and detection is done on a machine or an application level. Such tools, however, often lack knowledge of network context and can be bypassed by distributed low-profile attacks. Agent-based [2] and SIEM systems partially solve this problem by matching anomalies from end computers in one central point. They require specific software to be configured in end computers and thus fail with BYOD paradigm, though. Network-level monitoring gives a context and does not require any alteration in end computers.

We present four tools for network-level detection of dictionary attacks. *SSH-Monitor* detects SSH attacks by matching attack signatures [11]. It also comes with an observation that an attack is always preceded with a port scan and that most often two or more machines are targeted. *SSHCure* is based on an observation that an SSH attack is divided into three phases with distinct flow properties: the scanning phase, the brute-force phase and the die-off phase [4]. *RdpMonitor* is an extension of previous tools that is focused on a detection of RDP attacks [10]. *Honeyscan* uses synergy of honeypots and network monitoring to detect various dictionary attacks [5].

These tools, although each taking a different approach, share similar properties. They use thresholds, focus on statistical properties, and do not take into account the character of a traffic, i. e. variations in attack. Attackers could easily subvert these tools by changing their behavior, but practical experience shows they are not likely to. One reason is the supply of bruteforcing tools that in general operate on the same principle. They attack as much as a network or a user lets them and do not alter attack patterns. Their attacks are then very similar

and therefore detectable. In the terminology described next, they all belong to one traffic class.

3 Traffic Classes

A traffic class is a concept that tries to formally describe the character of an attack traffic by focusing on markers of automated action without special-casing for networks of different sizes. It is tailored for NetFlow, but it can be modified to work with e. g. packets. A basic primitive in this concept is a *slice* – a set of flows from one source IP address in a given time interval. This *slice* is characterized by four traffic dimensions.

The first traffic dimension is *visibility* that describes traffic in terms of newly created flows. It is quantized into two values that are functions of current defense capabilities of given network. Traffic in a *slice* is *visible* if there is at least one detection mechanism in a network that will mark it as anomalous/suspicious based on statistical properties of its flows. Traffic in a *slice* is *stealthy* otherwise. This way the dimension is relevant for networks of different sizes and detection capabilities.

The second dimension is *periodicity* that describes temporal relations between flows in a *slice*. It has three possible values. *Aperiodic* traffic has no discernible pattern in flow occurrence. *Periodic* traffic has a pattern in flow occurrence. *Constant* traffic has roughly the same number of flows in all *slices* of the same length.

The third dimension is *similarity* that deals with flow similitude. Flows in a *slice* can be *similar* or *different* based on arbitrary criteria. In this paper we use overall byte count and duration as a discriminant.

The fourth dimension is the *target count* that has two possible values: *one target* and *many targets*.

Dimensions that we introduced are orthogonal, so one value for each dimension can be chosen for every traffic. Therefore, any traffic can be described as a combination of these dimensions. Let any such combination of traffic dimensions be called the traffic class.

4 Relation Between Traffic Classes and Attacks

By combining all dimensions we get 24 traffic classes in total. However, they can be categorized into several groups depending on a dominant detectable dimension.

The first group is *noisy traffic* that consist of all *visible* combinations. Because traffic is considered *visible* if it was marked as an attack by at least one mechanism, this entire group of traffic classes is marked as an anomaly or an attack. Although this group covers one half of traffic classes, it does not necessarily cover half of attacks a network is facing. For example, a network with only a thresholding port scan detection would be able to detect periodic, aperiodic,

distributed, etc. scanners, but would fail to detect dictionary attacks, DoS or massive spamming.

The second group is *flat traffic* that consist of *stealthy* and *constant* combinations. This group covers traffic generated by bruteforcing tools as described in Section 2. Traffic with constant pattern that is disrupted only by network conditions and eventual proxy delays. However, not all flat traffic can be considered anomalous or harmful as there are valid use cases for such pattern, e. g. keep-alive. But, it is important to be aware that constant traffic is defined in terms of constant creation of new flows in given slice and not e. g. constant byte rate. Such constant appearance of new flows in case of authenticated protocols like SSH or RDP is always suspicious.

The third group is *spiked traffic* that consist of *stealthy* and *periodic* combinations. This group is a superset of *flat traffic* and it contains anomalies like beaconing [1], but also a lot of traffic that looks anomalous but in fact is not.

The fourth group is *episodic traffic* that consist of *stealthy* and *aperiodic* combinations. This group is interesting, because there is no discernible pattern in flow occurrence. Virtually nothing in the character of traffic points to whether it is an attack or not. Unless an attacker crosses some threshold, such attacks are undetectable on a network level.

5 Detecting Dictionary Attacks by Detecting Flat Network Traffic

In this section we present a proof-of-concept method that is built upon the assertion that *flat traffic* to authenticated services is always suspicious. We then analyze results of this method and compare it with other detection mechanisms running in the same network.

This method was developed and tested in the network of Masaryk University (about 15.000 devices online) in the course of two months from 24. 2. 2013 to 24. 4. 2013.

When designing the detection method, we have directly applied the definition of *constant* traffic that says that traffic is constant if all *slices* of the same length have roughly the same number of flows. However, we had to decide on three key characteristics – length of *slices*, their relative temporal position, and measure of their equal flow count. For the sake of brevity, we omit analyses behind our choices and present here only the final decisions. These analyses, however, are available upon request.

Our method works with 60 second slices of 5 minute batches of NetFlow data. These slices are adjacent and non-overlapping, i. e. there are at most 5 slices for one source IP address in a batch. *Slice* flow count equality is given by a function that measures the relative difference between flow count in adjacent non-zero *slices*. The method was successfully used with following protocols: FTP(S), SSH, Telnet, LDAP, HTTP(S), AFP, RDP and SMTP. *Flat traffic* detection for HTTP(S) was found to be ineffective, because valid AJAX updates common on Web 2.0 tend to produce flat traffic pattern.

5.1 Identification of Attackers

Figure 1 presents the number of identified instances of flat traffic in five-minute traffic windows and unique source IP addresses for various thresholds of relative difference. Only differences up to 109% was taken into account. Anything higher was automatically considered non-flat traffic. As was expected, graphs of identified source IP addresses grow faster than the graphs of attacks. In case of SSH and RDP services that are often targets of attacks, even the 50% threshold was enough to identify more than 90% of potential attackers.

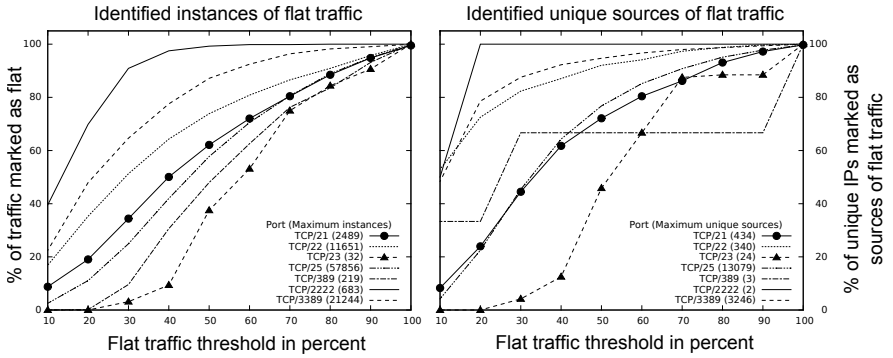

Fig. 1. Amount of flat traffic and its sources based on a relative difference threshold

Definitive evaluation of data is complicated, because the detection was carried out on an unannotated data, therefore, we have decided to compare our results with three other detection mechanisms deployed in the same network. We chose 60 % as a relative difference threshold and identified 3139 unique IP addresses targeting port TCP/3389. *RdpMonitor* [10] that detects attacks on this port found at the same time 852 attackers of which our method detected 410. Our method also identified 320 unique IP addresses targeting port TCP/22. *SSHMonitor* [11] found 29 attackers on this port of which we found 5, and *honeyscan* [5] found 57 attackers of which we found 15. Manual inspection of a sample of remaining addresses undetected by other tools concluded that they were likely to be actual attacks. However, conclusive proof is not available because the data on targeted machines is not available. The likely reason why the proof-of-concept method detects order of magnitude more attackers than deployed methods is their high threshold to avoid false-positives.

6 Conclusion

In this paper we have presented a model of attack traffic that puts emphasis on a distinction between artificial and human-initiated traffic. Based on this model, we created the proof-of-concept method that successfully detects one

type of attacks against various services. By applying concepts of this model, other methods can be devised that will detect other previously overlooked attacks that currently hide below detection thresholds. Relative simplicity of this model and its application also lowers bars for industry adoption.

Detection of malicious traffic based on a character of traffic proved to be a promising approach. In our future work, we will explore other detection schemes based on a character of traffic that can be used as a stand-alone or complementary detection mechanisms. We will also research other possible traffic classes and their respective groupings.

Acknowledgment This paper is based on a work supported by the Czech Ministry of Interior under Identification code VF2013201531.

References

1. Balland, P.: An Analysis of Network Beaconing Activity for Incident Response (2008) http://www.cert.org/flocon/2008/presentations/balland_flocon2008.pdf (retrieved online March 27, 2013)
2. Dasgupta, D., et al.: CIDS: An agent-based intrusion detection system. Computers & Security 24.5, pp. 387-398. (2005)
3. Dragon Research Group: SSH Password Authentication Report (2013), http://www.dragonresearchgroup.org/insight/sshpwauth.txt (retrieved online February 22, 2013)
4. Hellemons, L. and Hendriks, Luuk and Hofstede, R.J. and Sperotto, A. and Sadre, R. and Pras, A.: SSHCure: A Flow-Based SSH Intrusion Detection System. In: Proceedings of the 6th International Conference on Autonomous Infrastructure, Management, and Security (AIMS 2012), pp. 86-97, Springer Verlag, (2012)
5. Husák, M., Drašar, M.: Flow-based Monitoring of Honeypots. To appear in: Proceedings of 7th International Conference on Security and Protection of Information (SPI 2013)
6. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A., Rivest, R.L.: Identification and Entity Authentication. In: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
7. Seifert, C.: Analyzing Malicious SSH Login Attempts (2006), http://www.securityfocus.com/infocus/1876 (retrieved online March 27, 2013)
8. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B.: An Overview of IP Flow-Based Intrusion Detection. Communications Surveys Tutorials 12(3), 343356 (2010)
9. Thames, J.L., Abler, R., Keeling, D.: A Distributed Active Response Architecture for Preventing SSH Dictionary Attacks. In: IEEE Southeastcon 2008, pp. 8489 (2008)
10. Vizváry, M., Vykopal, J.: Flow-based Detection of RDP Brute-force Attacks. To appear in: Proceedings of 7th International Conference on Security and Protection of Information (SPI 2013)
11. Vykopal, J.: A Flow-Level Taxonomy and Prevalence of Brute Force Attacks. In: Advances in Computing and Communications, pp. 666-675, Springer Berlin Heidelberg, Berlin (2011)