Estimating Parameters of Target's Detection Methods

Martin Drašar, Jana Medková
Institute of Computer Science
Masaryk University
Brno, Czech Republic
{drasar,medkova}@ics.muni.cz

***Abstract*— Dictionary attacks are a prevalent phenomenon, which was lately amplified with the onset of insecure SOHO and IoT devices. Some of these devices use their own protection against dictionary attacks and some offload their security to mechanisms deployed in the infrastructure, such as flow-based IPS systems. These mechanism often operate with the notion of a typical attacker, who represent common attacks against the infrastructure and lacks sophistication. However, the emergence of stealthy and distributed dictionary attacks indicate that detection mechanisms should take sophisticated attackers into account. In this paper we explore the capability of a sophisticated attacker, who can estimate parameters of detection methods deployed on target and can then craft an attack which could go undetected for arbitrary long time. For this, we propose a new model of attacker-target interaction during dictionary attacks, which is based on attacker's perspective. Using this model we then postulate and experimentally evaluate an algorithm for estimating parameters of target detection method, illustrating that an attacker requires only a handful of attack attempts to correctly guess these parameters.**

I. INTRODUCTION

Despite introduction of various authentication methods, one-factor password-based authentication is still prevalent. Although home computers and servers are usually reasonably protected by local monitoring, there exist two groups of devices which gain notoriety for their lack of security and susceptibility to dictionary attacks: SOHO devices, such as routers, and IoT devices, as can be clearly demonstrated by existence of tools like Shodan. [1] Their widespread deployment, tendency to use default credentials and subsequent abuse in various botnets returns the issue of dictionary attacks back to forefront. Although many of these devices are, especially in a corporate environment, monitored by IDS/IPS solutions to offset their lack of security, these solutions expect a certain class of attackers with relatively visible behavior. However, as the emergence of stealthy and distributed attacks, and advanced persistent threats revealed, there are attacks that are predominantly stealthy and which can go unnoticed for a long time. [2] In our networks we have detected such long-running low-profile attacks, which convince us that they are commonly used and remain unchecked by many today systems.

In this paper we analyze the method for estimating parameters of target defense and for tailoring dictionary attacks which can go unnoticed with maximum efficiency. We hope that our research will motivate further research in detection of dictionary attacks and that the provided data will be used for better and more realistic evaluation of current and future detection methods.

This paper contributes to the state of the art in four ways:

- We present a simple model of interaction between an attacker and host or network based systems for detection of dictionary attacks. This model covers most currently available and deployed systems.
- We analyze available detection methods and derive a set of three classes of detection methods which are recognizable by an attacker and useful for parameter estimation.
- We present an algorithm for estimation of detection methods' parameters, which can be used for crafting attacks undetectable by target detection mechanism.
- We provide a tool for generation of datasets of detectors' behavior. This tool can be used to create training and testing datasets and also to evaluate efficiency of different attack strategies.

This paper is divided into six sections. The Section II surveys the state of the art in the area of host and network-based detection of dictionary attacks against common protocols and derives three classes of detection mechanisms. The Section III derives a model of interaction between attacker and host and network-based detectors based on six distinct features. The Section IV presents a tool for generation of detectors' datasets and for evaluation of attack strategies. The Section V introduces an algorithm for estimation parameters of detection methods, and presents its evaluation. The Section VI concludes the paper and outlines further research directions.

II. STATE OF THE ART

The majority of dictionary attacks target the following protocols: HTTP(S), SSH and RDP. The measures for preventing dictionary attacks can be divided into two groups: industrial and academic.

The industrial group is represented by host-based tools such as SSHGuard [3], fail2ban [4], denyhosts [5], or sshblack [6], which, after a predefined number of login attempts is reached, block the attacker by means of firewall configuration. Despite some of them having SSH in the name, they are useful for a number of protocols susceptible to dictionary attacks, such as IMAP, POP3, FTP, or HTTP. For securing web applications, there are many options to prevent or delay attacker, ranging from custom scripts, to large commercial solutions like Sucuri firewall [7] or Wordfence [8]. There

are also network-based industrial solutions, such as Flowmon ADS [9]

The academic group focuses mostly on network-based attack detection to alleviate the need for end machine management and to enable security of otherwise unsecurable devices. The SSH protocol is being the most common use-case. The approaches ranges from heuristics [10], statistics [11], detecting abnormal behavior using flat traffic detection [12], [13] or using hidden Markov models [14], to abnormalities in DNS traffic [15]. Recently, most work is focused on correctly distinguishing attack attempts in the network from legitimate traffic, e.g., [2] who detects stealthy dictionary attacks by measuring transition points of SSH protocol or [16], [17], who used machine learning for traffic classification. Additionally [14] presented an algorithm for detecting an actual compromise of an ssh service. From the point of defender, these academic approaches are all striving to make the detection more precise, with less false positives and less noise generated. From the point of an attacker, however, these enhancements make very little difference, unless they are actively trying to be stealthy.

Despite a large number of methods and tools, detection mechanisms can be divided into three types, depending on how they would react to the attack:

- immediate detection [e.g., SSHGuard, fail2ban]: After the attacker tries a predefined threshold of attempts, they are blocked.
- delayed flow-based detection [e.g., SSHCure, Flowmon ADS]: If the attacker tries a certain number of attempts in a given measured time window, they are blocked. Such evaluation is done in batches of typically five minutes.
- rate limiting [e.g., custom delay scripts]: The attacker faces delays in allowed attempts.

These types can be combined [e.g., fail2ban on host and flow-based solution on the network-level] and each of these groups have to be coupled with blocking mechanism to have any impact on the attacker.

III. MODEL OF HOST- AND NETWORK-BASED DETECTION MECHANISMS

In this section we present a model of detectors based on a number of assumptions, which are discussed in the text. As the aim of this paper is an estimation of detection methods' parameters, the model is built from the perspective of an attacker who has a number of unique attacking machines at hand. It is expected that every detection method is accompanied by a mechanism for blocking an attacker, regardless whether this block happens on the host or on the network level. Because this model is considered from the perspective of an attacker, a detection without action would be inconsequential for the attacker. It is also expected that a detector can differentiate each authentication attempt and that the attacker can discern target response correctly. This does not preclude modelling of detection methods which use technical means to fool automated attackers, but it leaves out the part that is largely implementation specific.

In this model, every detection method is described by the following six parameters:

- **Processing mode**: Either continuous or batch. The processing mode largely differentiates between flow-level network-based methods from the host-based ones. The flow-based methods usually process data in batches of a few minutes (processing window).
- **Response delay**: The delay between an attempt and any possible action. In the batch processing mode this represents the length of a processing window. In continuous processing mode this parameter is seldom used and can represent a processing delay caused by technical means (e.g., time required to update firewalls).
- **Time window**: A time span in which attacks are evaluated.
- **Blocking threshold**: The number of unsuccessful login attempts in a time window before block is issued.
- **Inter-attempt delay**: A delay between particular login attempts. The delay does not have to be a fixed number, it can, e.g., grow exponentially with each unsuccessful attempt.
- **Block duration**: A time required for attacker to be able to attempt another login on one target after being blocked.

An attacker and a target work in request-response fashion. An attacker initiates a login attempt and the target responds with one of the following:

- **Success**: A login attempt succeeded in trying an authentication. This does not mean that an attacker guessed the correct credentials, as this is an implementation detail depending on attacker's dictionary and target's authentication policy.
- **Delay**: A login attempt did not lead to authentication, because of being delayed.
- **Block**: A login attempt led to a blocking of the attacker.

Despite a target and its associated detector being two separate entities, as is in case of network-based detection, they act as one target with blocking capability.

In our model, an attacker interacts with a target only through authentication attempts and acquires all knowledge only from targets' responses. The attacker is thus able to collect the following statistics, which are later used for classification of detection methods and for estimating detection parameters:

- **Attempted intensity**: The number of login attempts per given timeframe, which the attacker tried.
- **Achieved intensity**: The number of login attempts per given timeframe, which the attacker managed (i.e., this number is lowered by delayed attempts).
- **Successful attempt count**: The number of successfully tried authentication attempts.
- **Block time**: Time elapsed before the attacker was blocked.

A. Limitations

The model as described has several limitations, which were considered and deemed as not considerably limiting

model's descriptive power.

The model cannot fully describe detection methods based on trend analyses, similarity search, etc. However, from the attacker's point of view, the specifics of these methods can be treated as an implementation detail, because on smaller time scales, even these detection methods can be expressed by this model.

The model currently considers only 1:1 relation between an attacker and a target. However, distributed attacks against an infrastructure with network-based detection indicate the need for N:M relation. This will very likely require an additive change to the model, but is unlikely to considerably impact the proposed detection parameters estimation. It is, however, one of future research directions.

The model does not consider attacks with large variations in intensity. However, given the applicability of constant/flat traffic characteristics as a basis for detection [13], [12], this limitation is not that important.

B. Examples

To give an example of an application of the model, we represent three detection methods: fail2ban [4], SSHCure [18], and a delaying web authentication script. The parameters are in Table I. Fail2ban was configured to block after 5 attempts in a day and to block for a day. SSHCure is processing data in five-minute windows and blocks if there are more than 20 attempts a minute in a time window. Because we are considering only a simple attacker, who attack with constant intensity, this translates to 100 attempts in the entire time window. The script allowed 10 attempts with exponential delays between them before blocking. Similitude between the model and real-life implementation was evaluated by comparing the results of the detection evaluation tool from the Section IV and the respective detection tools.

IV. DETECTION EVALUATION TOOL

We have developed a tool, which is able to build detection methods according to our model for evaluation. The tool is available at [19]. The tool consist of two types of objects: detectors and attackers. Detectors are created by supplying model parameters, attackers have to be implemented from scratch. However, we readily provide two types of attackers: a simple attacker which takes a number of attempts and a time window and then performs these attempts against a selected detector, and a sophisticated attacker which estimates detection parameters of selected detector and attempts to maximize the successful attempt count.

The tool can be used for two tasks: generating a detection dataset for a detector and direct evaluation of attacker and its strategy.

To generate our datasets, we ran an attacker with increasing intensity against particular detectors and gathered the statistics with attempted and achieved intensities, the successful attempt count, and a time to block. The scripts for generating these datasets are also a part of the repository.

To directly evaluate attackers' strategy, each attacker was given a number of lives, which represented the number of proxies the attacker can use. The attacker then made attempts against a detector, until it was blocked. After the block, if the attacker has some lives left, it subtracted life, changed the intensity and attacked again. The evaluation criteria were losing as little life possible and maximizing the number of attempts the attacker could do in span of fourteen days.

V. ESTIMATION OF DETECTION PARAMETERS

In order to optimize the attack, we need to know the exact type and parameters of the detector protecting the target. The estimates must be derived solely from the reactions of the detector. In this chapter we describe an algorithm, which determines the type and parameters of the detection by modifying the intensity of the attack.

A. Detection Method Type

The estimation of the detection method type is rather simple:

- if the achieved intensity is lower than attempted intensity, the detector exerts rate limiting
- if the successful attempt count is independent of the intensity and low, the detector exerts immediate detection
- otherwise the detector uses delayed flow-based detection.

B. Detection Method Parameters

We will now discuss how the parameters can be derived for each detection method type.

1) Rate Limiting: The parameters of the rate limiting method can be easily derived from the timing of the actual authentication attempts.

2) Immediate detection: The estimation of detection parameter for immediate detector is very simple. The only parameter is the successful attempt count, which remains constant through attacks.

3) Delayed flow-based detection: The estimation of delayed flow-based detection parameters is slightly more complicated. Three parameters of the model play a role:

- response delay: how often is the detection run (e.g., every 5 minutes),
- time window: the length of the interval over which the number of attempts is counted and compared to a detection threshold,
- blocking threshold: the number of attempts in a detection window, which is considered to be an indication of attack and a signal to block.

From the attacker's point of view, the most important is the ratio between blocking threshold and time window (detection intensity), because it specifies the maximal intensity of an undetected attack.

The only way to estimate the detection intensity is by trial and error. The attacker attacks the target with given intensity, waits for the detector's reaction, improves their estimate of the detection intensity and selects an intensity for the next attack until they are certain about the detection intensity. The goal is to choose the attacks' intensity so that the attacker minimizes the number of iterations to achieve

Method	Processing mode	Response delay	Time window	Blocking threshold	Inter-attempt delay	Block duration
Fail2ban	Continuous	0 sec.	86400 sec.	5	0 sec.	86400 sec.
SSHCure	Batch	300 sec.	300 sec.	100	0 sec.	86400 sec.
Delay script	Continuous	0 sec.	86400 sec.	10	Exponential	86400 sec.

TABLE I

MODEL PARAMETERS FOR DIFFERENT DETECTION METHODS

precise estimation of the detection intensity, as well as the number of times the attacker is detected during the iterations.

We propose an algorithm, that selects efficiently the attack intensity for next iteration based on the history of detector's reactions to previous attacks, so that both number of required attacks and detections are minimized.

Assume the detector has the time window of length W and the blocking threshold R and the detection intensity $I = R/W$. The algorithm bounds the area containing the actual combination (W, R) of time window and blocking threshold (Figure 1) and refines the bounds with each attack. The boundary consists of a lower and upper bound on the detection intensity (i_0, i_1), an upper bound on blocking threshold (r_0) and a lower bound for time window (w_0). Each step, the intensity of the next attempt i' is selected so that the bounded area is halved by the line $i'w$. The algorithm stops once the span between i_0 and i_1 is lower than the error level established in advance.

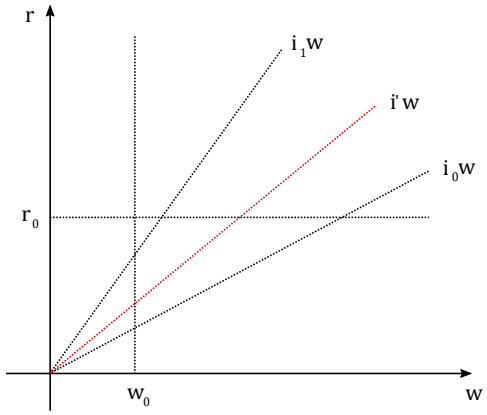

Fig. 1. Area Bounds

The algorithm improves the bounds as follows: assume we did an attack with given intensity i attempts per second. If the attack was not blocked, then the detection intensity must be greater than the intensity of the attack, therefore we can update the lower bound for the detection intensity

$$i_0^{k+1} = max(i_0^k, i).$$

On the other hand, if the attack was blocked after t seconds, following the same logic, we can update the upper bound on the detection intensity.

$$i_1^{k+1} = min(i_1^k, i)$$

Moreover, the total number of attempts in the attack is definitely higher than the threshold (can be actually much higher) and the time the detector needed to block the attack is not higher than twice the detection window. Therefore we can update the upper bound on the blocking threshold and the lower bound on the time window as follows:

$$r_0^{k+1} = min(r_0^k, i \cdot t),$$
$$w_0^{k+1} = max(w_0^k, \frac{t}{2}).$$

To evaluate the efficiency of our method, we compared it with the bisection method. The bisection method selects the intensity of the next attack as the average of the upper and lower bound on the detection intensity and updates the upper/lower bound when the attack is or is not blocked. We tested both methods on 180 combinations of blocking thresholds and time windows. The thresholds ranged from 10 to 40 with the step of 2. The time windows ranged from 5 seconds to 60 seconds with the step of 5 seconds.

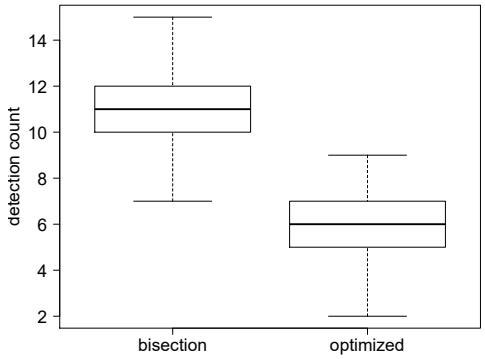

Fig. 2. Number of Detections

First, we compared the average number of times the attacker was detected by the detector. The overall results are shown in the Figure 2. The average number of detections for each step is shown in the Figure 3. Our method was detected fewer times during the process of estimating the detection method parameters than the bisection method. Since the attacker either needs a new proxy server or new attacking machine each time they are blocked by the detector, our method puts less strain on resources.

Another important property is how fast a method converges to the detection intensity. The Figure 4 summarizes how many attacks with different intensity the attacker needs to correctly estimate the detection intensity. The Figure 5 displays the speed of the convergence to the result. For each

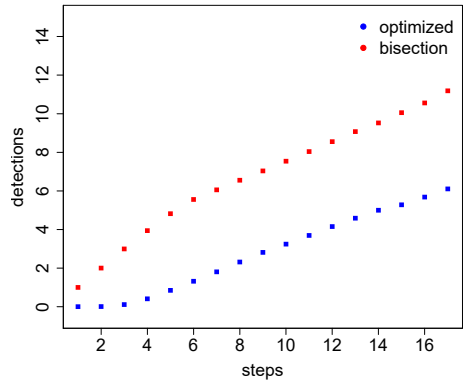

Fig. 3. Detections

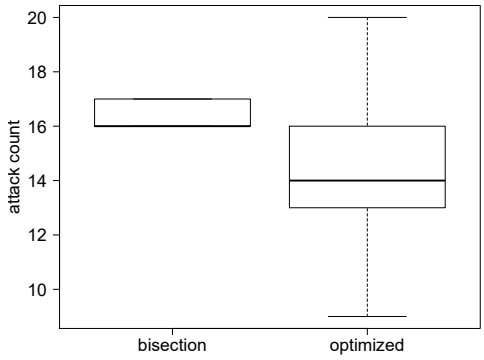

Fig. 4. Number of Attempts

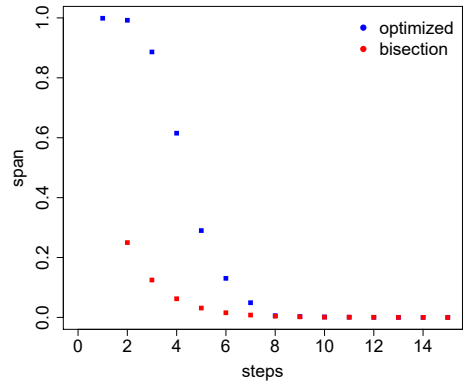

Fig. 5. Intensity Interval Span

step the average value of the span between the lower and upper bound is shown.

C. Combination of detection methods types

If the detector employs more than one detection methods types, the algorithm proceeds as follows:

- **rate limiting + immediate detection**: from the attacker's point of view, the immediate method is dominant. The rate limiting usually does not influence the achieved intensity for low number of attempts and the number of attempts is limited by the immediate method. In the rare case that rate limiting is triggered by a few attempts, the rate limiting parameters can be estimated from attempt timing.
- **rate limiting + delayed flow-based detection**: First, the rate limiting parameters are estimated from the timing of the attempts. For the purpose of estimating the delayed flow-based detection parameters, we can use the same algorithm with a minor modification, that the achieved intensity, not the attempted intensity, must be considered for limiting the area.
- **immediate detection + delayed flow-based detection**: Similarly to the combination of rate limiting and immediate, the delayed flow-based detection usually does not have any effect when combined with immediate detection, because it is always triggered before the delayed flow-based detection is triggered.
- **rate limiting + immediate detection + delayed flow-based detection**: see rate limiting + immediate detection and immediate detection + delayed flow-based detection.

VI. CONCLUSION

The increase in attackers' sophistication contests the typical notion of an attacker, which is prevalent in currently deployed and researched detection methods. In this paper we take on the role of a moderately sophisticated attacker and demonstrate, how such an attacker can craft dictionary attacks undetectable by target's detection mechanisms. To achieve this, we first present a short survey of currently used detection methods and demonstrate how, from the perspective of an attacker, all methods can be classified into one of three categories which dictate an attack strategy. We then present a model of attacker-target interaction during an attack, which is then used to derive an algorithm for precise estimation of parameters of target's detection mechanisms. With these information at hand, crafting an undetectable attack against arbitrary detection method is easy. To test capabilities of detection methods and to evaluate different attack strategies, we provide open access to a simulation tools which builds upon the presented model.

A. Future work

Despite the presented model's ability to capture current detection methods, it is still quite simple and not suitable for modeling more complicated attack scenarios and strategies. This include multiple attackers and multiple targets, and attacks with variable timing. In our future work, we want to provide such extension to this model and to evaluate it against deployed detection methods.

REFERENCES

[1] J. Matherly, "Shodan."
[2] A. Satoh, Y. Nakamura, and T. Ikenaga, "A flow-based detection method for stealthy dictionary attacks against secure shell," *Journal of Information Security and Applications*, vol. 21, pp. 31 – 41, 2015.
[3] "Sshguard." `https://www.sshguard.net/`. Accessed: 2017-07-01.

[4] "Fail2ban." https://www.fail2ban.org/. Accessed: 2017-07-01.
[5] "denyhosts." https://github.com/denyhosts/denyhosts. Accessed: 2017-07-01.
[6] "sshblack." http://www.pettingers.org/code/sshblack.html. Accessed: 2017-07-01.
[7] "Sucuri website application firewall." https://sucuri.net/website-firewall/. Accessed: 2017-07-01.
[8] "Wordfence." https://www.wordfence.com/. Accessed: 2017-07-01.
[9] "Flowmon ads." https://www.flowmon.com/en/products/flowmon/anomaly-detection-system. Accessed: 2017-07-01.
[10] J. Vykopal, *A Flow-Level Taxonomy and Prevalence of Brute Force Attacks*, pp. 666–675. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
[11] C. Callegari, S. Giordano, and M. Pagano, "New statistical approaches for anomaly detection," *Security and Communication Networks*, vol. 2, no. 6, pp. 611–634, 2009.
[12] M. Drašar, "Protocol-independent detection of dictionary attacks," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, pp. 304–309, Springer, Berlin, Heidelberg, 2013.
[13] M. Jonker, R. Hofstede, A. Sperotto, and A. Pras, "Unveiling flat traffic on the internet: An ssh attack case study," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 270–278, May 2015.
[14] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "Ssh compromise detection using netflow/ipfix," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 20–26, Oct. 2014.
[15] K. Takemori, D. Romaa, S. Kubota, K. Sugitani, and Y. Musashi, "Detection of ns resource record dns resolution traffic, host search, and ssh dictionary attack activities," *International Journal of Intelligent Engineering and Systems*, vol. 2, pp. 35–42, 12 2009.
[16] G. K. Sadasivam, C. Hota, and B. Anand, "Classification of ssh attacks using machine learning algorithms," in *2016 6th International Conference on IT Convergence and Security (ICITCS)*, pp. 1–6, Sept 2016.
[17] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, "Machine learning for detecting brute force attacks at the network level," in *2014 IEEE International Conference on Bioinformatics and Bioengineering*, pp. 379–385, Nov 2014.
[18] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, *SSHCure: A Flow-Based SSH Intrusion Detection System*, pp. 86–97. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
[19] "Tool for modeling attacker-target interaction." https://github.com/CSIRT-MU/Interactor3000. Accessed: 2017-07-01.