SDN-Assisted Network-Based Mitigation of Slow HTTP Attacks

Thomas Lukaseder, Lisa Maile, Frank Kargl
Institute of Distributed Systems
Ulm University, Germany
{firstname}.{lastname}@uni-ulm.de

***Abstract*—Slow HTTP attacks are hard to detect as the attackers behave according to the specification. The default configuration of most servers leaves them vulnerable to this attack. Meanwhile, the pressure to secure the attack targets shifts more and more to the network operators. Often without direct access to the target, the operators are asked to secure their clients. Software-defined networking (SDN) offers the flexibility and extensibility to analyze and influence network flows without help of the target operator. In previous work, we designed and built a framework based on software-defined networking and the Bro Network Security Monitor that can mitigate attacks within the network infrastructure without access to the attack target. The initial framework can reliably mitigate different flooding attacks. The presented project discusses strategies to add mitigation of slow HTTP attacks to this framework.**

I. Introduction

Distributed Denial of Service (DDoS) attacks can cause large-scale damage to service providers on the internet. Many smaller server operators do not have the resources or the knowledge to defend against these attacks and — depending on the attack — often do not have the overview of the network necessary to mitigate attacks effectively. SDN offers the unique opportunity to build a system that can observe the network status while simultaneously being able to actively intervene in the network structure to mitigate the attacks. Therefore, we introduced an extensible network-based DDoS mitigation framework based on SDN to mitigate attacks without support of the server operators [1][1].

The system is capable of mitigating flooding attacks reliably; other attacks are still work in progress. The system observes servers and recognizes attacks as a first step. In a second step, if an attack was discovered, the attackers are identified by observing the incoming traffic of the server. Therefor, the controller directs the SDN-assisted OpenFlow switch to mirror all incoming packets of the server to the mitigation system. Every client is then assigned a score depending on the load it generates on the server. In a third step, the OpenFlow switch redirects the clients with the highest scores to a challenge server to identify bots and limits their data rate or blocks them from accessing the web server. Thereby, SDN allows the complete automation of all redirecting, limiting and blocking tasks of this framework.

[1]The corresponding paper is already accepted but not yet published. Please contact the authors to receive a copy.

The first step is attack independent and therefore also works for slow HTTP attacks. However, in the second step, the behavior of slow HTTP attackers differs greatly from other attack types and the scoring system cannot detect these attacks in the current setup. Some additional changes are also necessary for the third step, as the connections have to be closed.

II. Slow HTTP Attacks

In contrast to most DDoS attacks, slow HTTP attacks do not generate a large amount of packets and use only a marginal amount of bandwidth. They aim to use resources on the server by forcing it to remember connections as long as possible with a minimal amount of work by the client. The slow header attack or Slowloris is the most well-known slow HTTP attack. It uses fragmented headers to occupy the targeted servers resources for as long as possible. An attacker sends the first header with only one CRLF at the end to indicate that the header is not finished. Therefore, the server assumes that the missing header is due to a slow internet connection of the client and keeps the connection open. The slow body attack — also called slow POST attack — is similar. However, it uses complete HTTP POST request headers but sends the body as slowly as possible. Servers wait for a configured timeout after the last received packet before they close the connections. With most default configurations, attackers can wait nearly 5 minutes before sending the next packet to a server.

Slow HTTP attacks can occur in different manifestations. If the attack is distributed, it is sufficient for each attacking client to establish a very low amount of connections with only very few packets. On the other hand, the high efficiency of the attack also allows non-distributed attacks with only one attacker which then has to open many connections and has to send a rather large amount of packets.

III. Attacker Identification

Only a small amount of research has been conducted with regard to network-based slow HTTP attack detection. Intrusion detection systems are not able to successfully distinguish these attacks from normal server traffic as the attack does not contain any malformed requests [2].

Currently, many servers such as Apache can be configured to mitigate the effect of slow HTTP attacks by limiting the maximum time a server waits to receive a full request.

However, these changes also block legitimate requests from clients with slow internet connections and an attack still has a noticeable impact on the server's performance [3]. This mitigation technique requires the administrator to become active and is therefore not a viable option for our use case.

In our framework, the attacker identification process begins after an attack on the server is detected. In this phase, the communicating clients are scored according to their sent packets. Packets that result in a high effort for the server are rated with a higher suspiciousness value than packets which result only in small memory and CPU usage. Because of the low traffic rate of slow HTTP attacks, and therefore small number of packets from each attacker, this way to rate clients needs to be extended and adapted. Based on our research of possible ways in which these attacks can occur, we identified different techniques to extract attackers that are evaluated in this project:

a) Low and evenly distributed packet rate: An easy way of identifying and scoring clients is by testing whether consecutive packets of the same connection have an evenly distributed time period between them. If this behavior is identified, the clients are assigned a high suspiciousness score as this implies, that the packets are not sent slowly because of a bad connection but deliberately. In a more generic approach, connections with a significantly low packet rate are rated as suspicious. For the optimal implementation of this method, the first messages which belongs to the TCP handshake should be ignored and the bandwidth metrics are only calculated with the remaining packets. More precise approaches that take the changes of the sending rate into account lead to a high management effort since every characteristic of each connection needs to be remembered.

b) Long connections: Another simple method measures the duration of connections and assigns suspiciousness scores for very long connections. This method, however, needs to wait for the connection to last longer than a certain threshold in the area of minutes and, thus, the identification of attackers can take a long time.

c) The amount of incomplete packets: This method helps to identify attackers reliably since slow HTTP attacks always require a significantly high amount of incomplete packets. The identification of incomplete packets needs to check GET messages for only one end of line character at the end or compare the content-length definition with the actual body length of POST messages. This identification method is quite resource intensive as deep packet inspection is necessary. However, it can still be evaluated if this method would increase the accuracy of the attacker identification.

d) Backup mechanisms: Lastly, slow POST attacks can be identified if POST messages are assigned a very high suspiciousness score. To prevent non-distributed DoS attacks, the number of connections per single clients can be considered. Thereby, any additional connection will increase the score which is assigned to a client. If other methods fail to identify attackers correctly, these two mechanisms can at least provide the support for a small subgroup of slow HTTP attacks.

All techniques include high management effort to remember connection information, calculate suspiciousness scores and connection characteristics. It remains to be seen if the described techniques are efficient enough to be handled by the framework or if additional resources such as a bigger cluster, a pre-filter or mathematical models such as probabilistic counting are necessary. However, even with possibly open scalability issues, this project can evaluate the different techniques in terms of practicability and precision.

IV. Attack Mitigation

In the last phase, called mitigation phase, the clients with the highest suspiciousness score are considered attackers and redirected to a CAPTCHA server or blocked. Thereby, it is assumed that most DDoS attacks are done by bots and, thus, the attacker clients are not able to solve the CAPTCHA challenge and will remain blocked. If the clients are just redirected and further messages are blocked via the SDN-assisted Switch, the server keeps the connections open until the default timeout is reached. This is acceptable for flooding attacks, as the amount of open connections is not the resource under attack. For slow HTTP attacks however, this could lead to an inefficient mitigation and will further increase the downtime of the server even after the attackers are identified. Therefore, it is necessary to send TCP-RST-messages from the protection system to the server. For these RST-messages, the sequence- and acknowledgement-number of the connections need to be extracted which again results in considerable effort.

V. Conclusion

Slow HTTP attacks differ quite a bit from flooding attacks and their identification and mitigation might lead to a high management effort of the network infrastructure. We developed several concepts based on flow-based analysis of network traffic that can help identify attackers and how to exclude them from the network. Preliminary deliberations suggest that scalability of the network data analysis could be an issue. With slow HTTP attacks, we are faced with the necessity to analyse large amounts of data efficiently. Performance and accuracy of the mitigation system are in sharp contrast to each other and the right middle ground needs to be found.

Acknowledgment

This work was supported in the bwNET100G+ project by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK). The authors alone are responsible for the content of this paper.

References

[1] T. Lukaseder, A. Hunt, C. Stehle, D. Wagner, R. van der Heijden, and F. Kargl, "An Extensible Host-Agnostic Framework for SDN-Assisted DDoS-Mitigation," in *Proceedings of the 42nd Conference on Local Computer Networks (accepted)*, 2017.

[2] J.-B. Voron, C. Démoulins, and F. Kordon, "Adaptable Intrusion Detection Systems Dedicated to Concurrent Programs: A Petri Net-Based Approach." Washington, DC, USA: IEEE Computer Society, 2010, pp. 57–66.

[3] D. Moustis and P. Kotzanikolaou, Eds., *Evaluating Security Controls Against HTTP-based DDoS Attacks*. Fourth International Conference on Information Intelligence Systems and Applications (IISA), 2013.