Poster: Botnet over Slack: The Abusing of APIs in Third-party Chat Platforms

Fangjiao Zhang[1,2], Xiang Cui[3,1], Di Wu[1,2], Jianjun Zhao[1]
1 (Institute of Information Engineering, Chinese Academy of Sciences)
2 (School of Cyber Security, University of Chinese Academy of Sciences)
3 (Cyberspace Institute of Advanced Technology, Guangzhou University)
zhangfangjao@iie.ac.cn

***Abstract*—Botnets, as one of the most effective platforms to launch cyber attacks, remain the concern for researchers. However, most of existing command-and-control (C&C) channels are easily detected and traced with advanced technology. As such, we propose a C&C channel, which exploits Slack's application programming interface(API) for communication. Slack is a popular third-party chat platform, including API components that allow users to integrate their apps onto platforms themselves. API refers to a list of predetermined commands. They are not only vital to how Slack functions, but also to how attackers can abuse them. For the botnet over Slack we introduced, on one hand, there is no need to create a server for malicious activities. On the other hand, it's hard to detect and trace attackers using such a legitimate service. Our preliminary experimentation demonstrates the botnet has better robustness and concealment.**

Keywords—botnet; slack; abusing; API; third-party chat platforms

I. Introduction

Malware is still a critical security threat across the Internet. And as one of the most effective platforms to launch cyber-attacks, botnets are compromised of vulnerable end hosts remotely controlled by botmasters via C&C channels. They are widely used for malware distributing, ransomware distributing, spamming, DDoS, phishing, massive online identity theft and digital currency mining and so on. They are posing a major threat to the cyberspace and causing extensive concerns of the international. Accordingly, the researchers have raised a great deal of approaches for detecting botnets, analyzing based on either C&C channels used by botmasters to communicate with each infected machine or patterns of crowd behaviors exhibited by collections of bots in response to botmaster commands [1]. Over the years, various techniques have been used to set up C&C servers. However, most of existing C&C channels are not quite robust and resilient, and also are easily traced.

Slack is one of the most popular online third-party chat platforms for inter-office communications. It integrates with many different apps and services people already use every day. The Slack company has been established for just more than 4 years, but with 5.8 million active accounts and a valuation of 9 billion dollars. In fact, 77% of fortune 100 companies reported use it driven by the high degree of practicality. By using the Slack's API, people could integrate core chat services with custom apps that they could access. In other words, an employee could simply use one app for everything rather than switching between different softwares. For instance, Twitter is interfaced with Slack, so one can receive notifications of upcoming messages from twitter, and not have to access the website *http://www.twitter.com* for it. Despite of the convenience and efficiency achieved by using APIs, they also introduces vulnerabilities to some extent and could be used to create C&C infrastructure to control malware-infected hosts for the attackers remotely performing malicious activities.

Considering the feasibility, in this paper, we design a botnet over Slack and make the following three contributions. Firstly, we introduce a C&C channel utilizing APIs in the chat platform Slack, in which the botmaster and bots could communicate with each other via valid API requests. Secondly, the malicious traffic is mixed in with the normal traffic by using the same domain name slack-msgs.com. Therefore, it's hard to distinguish one from the other. And it's impossible to block the domain just for the sake of security. Besides, the HTTPS protocol is used by the instance of Slack. The above both increase the stealth of the malicious. Lastly, we use public services as part of the C&C channel to store command addresses and execution results of commands, such as URL shortening service(USS), Online Clipboard or online storage service. There is no need for attackers to deploy servers. And it's hard to be completely shut down [2].

II. Design

Assuming we have a group of compromised devices as bots, and each bot has the ability to reach the Internet. Here we mainly discuss the message delivery between the botmaster and bots by the C&C channel we design. Symmetric and asymmetric cryptography are used for information encryption and identity authentication to prevent from monitoring and hijacking, which is very mature in botnet construction. So we won't repeat them in the paper. The command and control process of botnet (Figure 1) mainly consists of three parts: **address**, **issue commands** and **receive results**.

A. Address ("①" and "②" in Figure 1)

Instead of issuing commands directly to bots, we store addresses of commands in popular public services (e.g. USS) widely used in the daily lives. Aliases or URLs could be customized by the public services so that the botmaster and bots share a series of addresses by means of the same address generation algorithms. We would prepare a default list of public services for the botnet. The botmaster and bots could use any service in it to exchange information. The botmaster would randomly select some services in the list and then use the address generation algorithm to generate an address for each service. Correspondingly, the bot also chooses random services on a regular basis, using the same algorithm to generate a series of addresses. Bot would access the above addresses and try to fetch the content. If the content exists and can be successfully decrypted, then it proves that the address is correct further getting command addresses naturally.

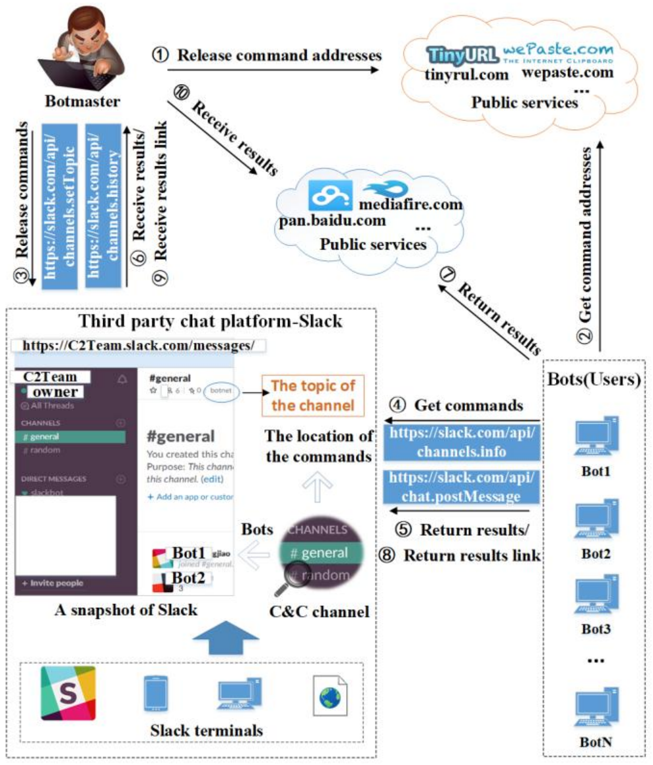

Fig. 1. The process of botnet command and control.

B. *Issue commands (“③” and “④” in Figure 1)*

Real commands the botmaster send to bots are stored in Slack. A team (e.g. C2Team) should be registered for employees in a company to communicate. Across the team, different channels are made for a project, a topic or anything, making sure the right people are involved at the right time. We select the default channel *#general* as the C&C place. There are many choices for the botmaster to store commands in the channel, such as *topic*, *profile*, *pinned items*, *shared files*, *threading messages* and so on. We select *topic* as the channel to issue commands by invoking the API *channels.setTopic*. Botmaster could periodically update the content on a customizable set of intervals. Then bots would obtain commands by API *channels.info* containing the *topic* of the channel.

C. *Receive results (“⑤” - “⑩” in Figure 1)*

After executing defined commands, bots would return results to botmaster. Due to the free version’s 5GB total file storage limit, there are two ways for bots to transmit results to the botmaster. If the size of results are small and there are not any files needed to upload, bots could send results to channel directly by calling API *chat.postMessage*(⑤-⑥). Then the botmaster uses API *channels.history* to retrieve results in *#general*, designed to fetches history of messages from the channel. Otherwise, the public data synchronization services, such as online storage services and sync notes, would be used as temporary storage places. Bots would firstly upload results to public services using APIs of their own(⑦) and then send the link address specified in a message to the channel(⑧). And the botmaster could get the results address from the channel and download results from the corresponding public service(⑨−⑩).

III. PRELIMINARY RESULTS

To prove the feasibility of the botnet we propose, we investigate some available public services used in the botnet. We choose and verify some of them for releasing command addresses and storing results from bots. So far, there are forty-six online storage services, eleven internet clipboards and eight social networks we have been counting so far. There are still other categories could be used in the botnet we should be working on. Besides, we realize the botnet over Slack by using its APIs. The Slack itself also provides the possibility of the botnet. For example, multiple simultaneous logins are allowed without any warning. Experiments show that our idea is feasible and has a higher concealment when programmed in *powershell* in practice.

IV. CONCLUSION & COUNTERMEASURES

Third-party chat platforms like Slack, Discord and Telegram are user-friendly and becoming more and more-popular owing to their free, convenient, externally hosted apps. However, they are also an idle alternative C&C medium for attackers. Telegram APIs are abused by *Telebots* group to receive commands and send responses in the December 2016 attack on Ukraine’s power grid. This is an area that we have to pay attention to. Apart from restricting users’ registration and login, suppliers should regulate the use of APIs and adopt a stronger authentication for communication for less abuse.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments for improving this paper. This work is supported by National Natural Science Foundation of China under grant (No. 61702508, No. 61602470), the Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences and Beijing Key Laboratory of Network Security and Protection Technology.

REFERENCES

[1] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, C. Kruegel, Disclosure: Detecting Botnet Command and Control Servers Through Large-scale NetFlow Analysis. Proceedings of the 28th Annual Computer Security Applications Conference, 129–138 (2012).

[2] Jianjun Zhao, Fangjiao Zhang, Chaoge Liu. Poster: ServerLess C&C channel. In poster Session of the 38th IEEE Symposium on Security & Privacy.2017.