ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

Spy Bot- Bot Net Detection

Raagavi P, Palaniyapan
Saveetha School of Engineering,Thandalam

Manuscript Info

Manuscript History:

Received: 15 December 2014
Final Accepted: 22 January 2015
Published Online: February 2015

Key words:

Corresponding Author

Raagavi.p

Abstract

Botnets are now recognized as one of the most serious security threats. In contrast to previous malware, botnets have the characteristic of a command and control (C&C) channel. Botnets also often use existing common protocols,
e.g., IRC, HTTP, and in protocol-conforming manners. This makes the detection of botnet C&C a challenging problem. In this paper, we propose an approach that uses network-based anomaly detection to identify botnet C&C channels in a local area network without any prior knowl- edge of signatures or C&C server addresses. This detection approach can identify both the C&C servers and infected hosts in the network. Our approach is based on the observa- tion that, because of the pre-programmed activities related to C&C, bots within the same botnet will likely demonstrate spatial-temporal correlation and similarity. For example, they engage in coordinated communication, propagation, and attack and fraudulent activities. Our prototype system,SpyBot, can capture this spatial-temporal correlation in network traffic and utilize statistical algorithms to detect botnets with theoretical bounds on the false positive and false negative rates. We evaluated SpyBot using many real- world network traces. The results show that SpyBot can detect real-world botnets with high accuracy and has a very low false positive rate.

Introduction

Botnets (or, networks of zombies) are recognized as one of the most serious security threats today. Botnets are different from other forms of malware such as worms in that they use command and control (C&C) channels. It is important to study this botnet characteristic so as to develop effective countermeasures. First, a botnet C&C channel is relatively stable and unlikely to change among bots and their variants. Second, it is the essential mechanism that allows a “botmaster” (who controls the botnet) to direct the actions of bots in a botnet. As such, the C&C channel can be considered the weakest link of a botnet. That is, if we can take down an active C&C or simply interrupt the communication to the C&C, the botmaster will not be able to control his botnet. Moreover, the detection of the C&C channel will reveal both the C&C servers and the bots in a monitored network. Therefore, understanding and detecting the C&Cs has great value in the battle against botnets.

Many existing botnet C&Cs are based on IRC (Internet Relay Chat) protocol, which provides a centralized command and control mechanism. The botmaster can interact with the bots (e.g., issuing commands and receiving responses) in real-time by using IRC PRIVMSG messages. This simple IRC-based C&C mechanism has proven to be highly successful and has been adopted by many botnets. There are also a few botnets that use the HTTP protocol for C&C. HTTP-based C&C is still centralized, but the botmaster does not directly interact with the bots using chat-like mechanisms. Instead, the bots periodically contact the C&C server(s) to obtain their commands. Because of its

proven effectiveness and efficiency, we expect that centralized C&C (e.g., using IRC or HTTP) will still be widely used by botnets in the near future. In this paper, we study the problem of detecting centralized botnet C&C channels using network anomaly detection techniques. In particular, we focus on the two commonly used botnet C&C mechanisms, namely, IRC and HTTP based C&C channels. Our goal is to develop a detection approach that does not require prior knowledge of a botnet, e.g., signatures of C&C patterns including the name or IP address of a C&C server. We leave the problem of detection of P2P botnets (e.g., Nugache [19], and Peacomm [14]) as our future work

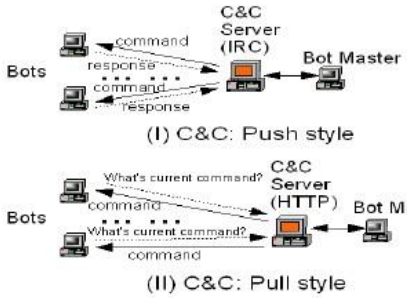
In a pull style C&C, the botmaster simply sets the

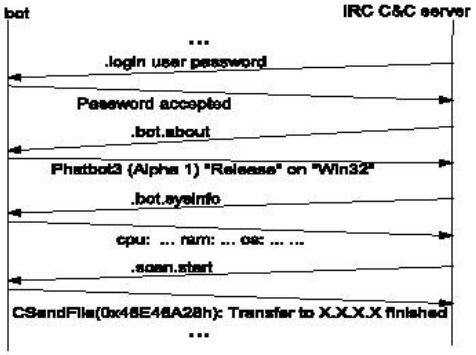
Botnet C&C traffic is difficult to detect because: (1) it follows normal protocol usage and is similar to normal traffic, (2) the traffic volume is low, (3) there may be very

few bots in the monitored network, and (4) may contain encrypted communication. However, we observe that the bots of a botnet demonstrate spatial-temporal correlation and similarities due to the nature of their pre-programmed response activities to control commands. This invariant helps us identify C&C within network traffic. For in- stance, at a similar time, the bots within a botnet will execute the same command (e.g., obtain system informa- tion, scan the network), and report to the C&C server with the progress/result of the task (and these reports are likely to be similar in structure and content). Normal network activities are unlikely to demonstrate such a synchronized or correlated behavior. Using a sequential hypothesis testing algorithm, when we observe multiple instances of corre- lated and similar behaviors, we can conclude that a botnet is detected.

2 Botnet C&C Architecture

As shown in Figure 1(a), centralized C&C architecture can be categorized into “push” or “pull” style, depending on how a botmaster’s commands reach the bots. In a push style C&C, the bots are connected to the C&C server, e.g., IRCserver, and wait for commands from botmaster. The botmaster issues a command in the channel, and all the bots connected to the channel can receive it in real-time. That is, in a push style C&C the botmaster has real-time control over the botnet. IRC-based C&C is the representative example of push style. Many existing botnets use IRC, including the most common bot fami-lies such as Phatbot, Spybot, Sdbot, Rbot/Rxbot, GTBot [5]. A botmaster sets up an (or a set of) IRC server(s) as C&C hosts. After a bot is newly infected, it will connect to the C&C server, join a certain IRC channel and wait for commands from the botmaster. Commands will be sent in IRC PRIVMSG messages (like a regular chatting message) or a TOPIC message. The bots receive com- mands, understand what the botmaster wants them to do,and execute and then reply with the results. Figure 1(b) shows a sample command and control session. The botmas- ter first authenticates himself using a usern ame/password.Once the password is accepted, he can issue commands to obtain some information from the bot. For example, “.bot.about” gets some basic bot information such as version. “.sysinfo” obtains the system information of the bot infected machine, “.scan.start” instructs the bots to begin scanning for other vulnerable machines. The bots respond to the commands in pre-programmed fashions. The botmaster has a rich command library to use [5], which enables the botmaster to fully control and utilize the in-fected machines.

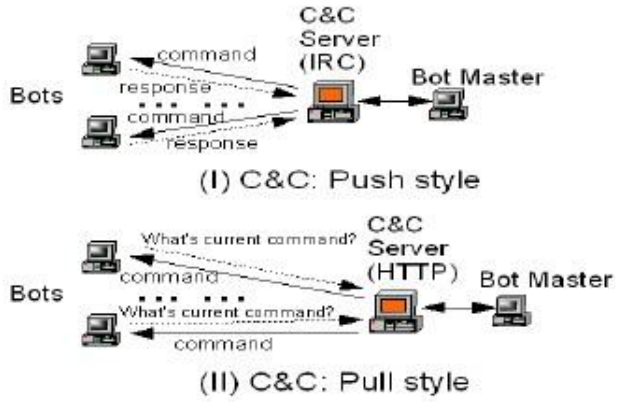


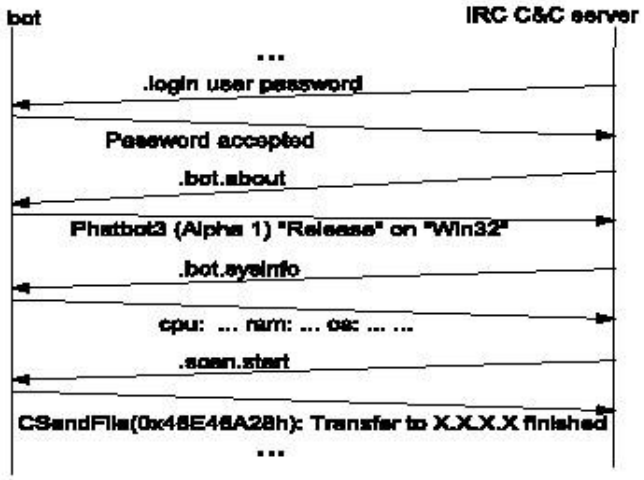


The bots frequently connect back to read the command file. This style of command and control is relatively loose in that the botmaster typically does not have real-time control over the bots because there is a delay between the time when he "issues" a command and the time when a bot gets the command. There are several botnets using `HTTP` protocol for C&C, for example, Bobax [25], which is designed mainly to send spams. The bots of this botnet periodically connect to the C&C server with an `URL` such as `http://hostname/reg?u=[8-digit-hex-id] &v=114`, and receive the command in a `HTTP` response.

As shown in Figure 1(a), centralized C&C architecture can be categorized into "push" or "pull" style, depending on how a botmaster's commands reach the bots. In a push style C&C, the bots are connected to the C&C server, e.g., IRCserver, and wait for commands from botmaster. The botmaster issues a command in the channel, and all the bots connected to the channel can receive it in real-time. That is, in a push style C&C the botmaster has real-time control over the botnet. IRC-based C&C is the representative example of push style. Many existing botnets use IRC, including the most common bot fami-lies such as Phatbot, Spybot, Sdbot, Rbot/Rxbot, GTBot [5]. A botmaster sets up an (or a set of) IRC server(s) as C&C hosts. After a bot is newly infected, it will connect to the C&C server, join a certain IRC channel and wait for commands from the botmaster. Commands will be sent in IRC PRIVMSG messages (like a regular chatting message) or a TOPIC message. The bots receive com- mands, understand what the botmaster wants them to do, and execute and then reply with the results. Figure 1(b) shows a sample command and control session. The botmas- ter first authenticates himself using a username/password. Once the password is accepted, he can issue commands to obtain some information from the bot. For example, ".bot.about" gets some basic bot information such as version. ".sysinfo" obtains the system information of the bot infected machine, ".scan.start" instructs the bots to begin scanning for other vulnerable machines. The bots respond to the commands in pre-programmed fashions. The botmaster has a rich command library to use [5], which enables the botmaster to fully control and utilize the in-fected machines.

command in a file at a C&C server (e.g., a `HTTP` server).





The command is in one of the six types, e.g., `prj` (send spams), `scn` (scan others), `upd` (update binary). Botnets can have fairly frequent C&C traffic. For example, in a CERT report [16], researchers report a Web based bot that queries for the command file every 5 seconds and then executes the commands. 2.1 Botnet C&C: Spatial-Temporal Correlation and Similarity There are several invariants in botnet C&C regardless of the push or pull style. First, bots need to connect to C&C servers in order to obtain commands. They may either keep a long connection or frequently connect back. In either case, we can consider that there is a (virtually) long-lived session of C&C chan- nel.1 Second, bots need to perform certain tasks and respond to the received commands. We can define t wo t ypes of responses observable in network traffic, n amely, message response and activity response. A typical example of mes- sage response is `IRC-based PRIVMSG` reply as shown in Figure 1(b). When a bot receives a command, it will execute and reply in the same `IRC` channel with the execution result (or status/progress). The activity responses are the network activities the bots exhibit when they perform the malicious tasks (e.g., scanning, spamming, binary update) as directed by the botmaster's commands. According to [31], about 53% of botnet commands observed in thousands of real- world `IRC-based` botnets are scan related (for spreading or DDoS purpose), about 14.4% are binary download related (for malware updating purpose). Also, many `HTTP-based` botnets are mainly used to send spams [25]. Thus, we will observe these malicious activity responses with a high probability [8]. If there are multiple bots in the channel to respond to commands, most of them are likely to respond in a similar fashion. For example, the bots send similar message or activity traffic at a similar time window, e.g., sending spam as in [23]. Thus, we can observe a response crowd of botnet members responding to a command, as shown in Figure 2. Such crowd-like behaviors are consistent with all botnet C&C commands and throughout the life-cycle of a botnet. On the other hand, for a normal network service (e.g., an `IRC` chatting channel), it is unlikely that many clients consistently respond similarly and at a similar time. That is, the bots have much stronger (and more consistent) synchronization and correlation in their responses than nor- mal (human) users do.

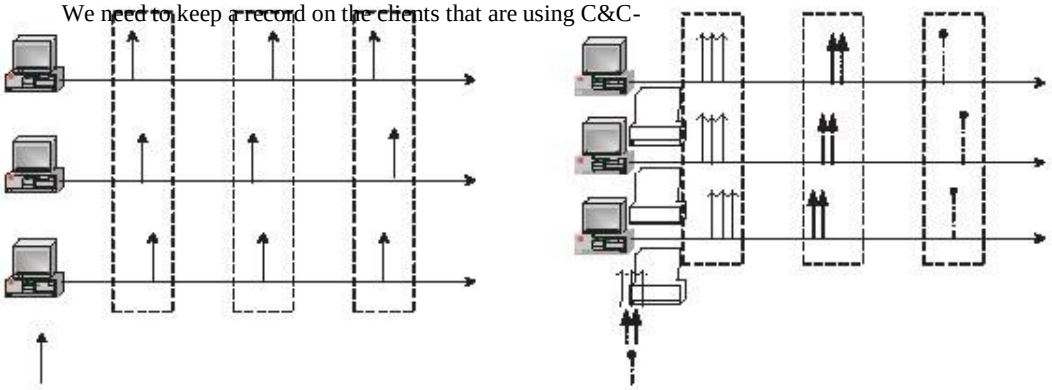
Based on the above observation, our botnet C&C detec- tion approach is aimed at recognizing the spatial-temporal correlation and similarities in bot responses. When monitor-ing network traffic, as the detection system observes multi- ple crowd-like behaviors, it can declare that the machines in the crowd are bots of a botnet when the accumulated degree of synchronization/correlation (and hence the likelihood of bot traffic) is above a given threshold.

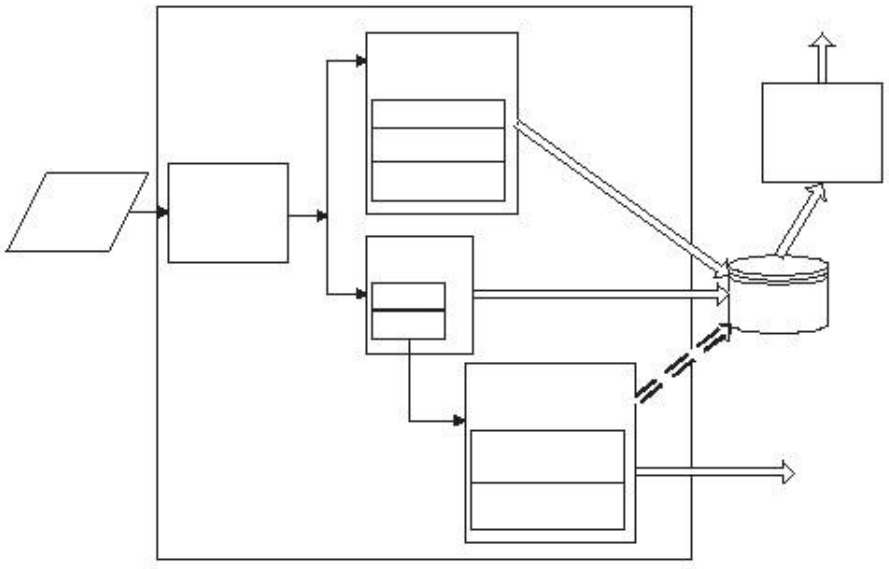
3 Botnet Detector:

Figure 3 shows the architecture of SpyBot. There are two main components, i.e., the monitor engine and thecorrelation engine. The monitor engine is deployed at the perimeter of a monitored network. It examines etworktraffic, generates connection record of suspicious C&C pro- tocols, and detects activity response behavior (e.g., scan- ning, spamming) and message response behavior (e.g., `IRC PRIVMSG`) in the monitored network. The events observed by the monitor engine are analyzed by the correlation en-gine. It performs group analysis of spatial-temporal cor- relation and similarity of activity or message response be- haviors of the clients that connect to the same `IRC` or `HTTP` server. We implemented the monitor engines as several preprocessor plug-ins on top of the open-source system Snort [24], and implemented the correlation engine in Java. We also implemented a real-time message response correlation engine (in C), which can be integrated in the monitor engine. The monitor engines can be distributed on several networks, and collect information to a central repository to perform correlation analysis. We describe each SpyBot component in the following sections

3.1.2 **C&C-like** Protocol Matcher

We need to keep a record on the clients that are using C&C-





A soft whitelist is generated for those addresses declared "normal" in the analysis stage, i.e., those clearly declared "not botnet". The difference from a hard list is that a soft list is dynamically generated, while a soft white address is valid only for a certain time window, after which it will be removed from the list. For activity response detection, SpyBot can monitor all local hosts or a "watch list" of local clients that are using C&C-like protocols. The watch list is dynamically updated from protocol matchers. The watch list is not required, but if one is available it can improve the efficiency of SpyBot because its activity response detection component only needs to monitor the network behaviors of the local clients

like protocols for correlation purpose. Currently, we focus on two most commonly used protocols in botnet C&C, namely, IRC and HTTP. We developed port-independent protocol matchers to find all suspicious IRC and HTTP traffic. This port-independent property is important because many botnet C&Cs may not use the regular ports. Wediscuss in Section 5 the possible extensions. IRCand HTTPconnections are r elatively simple to rec- ogn ize. For example, an IRC session begins with con-nection registration (defined in RFC1459) that usually has three messages, i.e., PASS, NICK, and USER. We can easily recognize an IRC connection using light-weight payload inspection, e.g., only inspecting the first few bytes of the payload at the beginning of a connection. This is similar to HiPPIE [1]. HTTP protocol is even easier to recognize because the first few bytes of a HTTP request have to be "GET", "POST", or "HEAD".
on the list.

3.1.3 Activity/Message Response Detection

For the clients that are involved in IRC or HTTP com-munications, SpyBot monitors their network activities for signs of bot response (message response and activity response). For message response, SpyBot monitors the IRC PRIVMSG messages for further correlation analysis. For scan activity detection, SpyBot uses approaches sim-ilar to SCADE (Statistical sCan Anomaly Detection En-gine) that we have developed for BotHunter [15]. Specif-ically, SpyBot mainly uses two anomaly detection mod-ules, namely, the abnormally high scan rate and weighted failed connection rate. SpyBot uses a new detector for spam behavior detection, focusing on detecting MX DNS query (looking for mail servers) and SMTPconnections (be-cause normal clients are unlikely to act as SMTP servers). We note that more malicious activity response behaviors can be defined and utilized in SpyBot. For example, binary downloading behavior can be detected using the same approach as the egg detection method in BotHunter [15].

3.2 Correlation Engine

In the correlation stage, SpyBot first groups the clients according to their destination `IP` and port pair. That is, clients that connect to the same server will be put into the same group. SpyBot then performs a group analysis ofspatial-temporal correlation and similarity. If SpyBot detects any suspicious C&C, it will issue botnet alerts. In the current implementation, SpyBot uses the Response-Crowd-Density-Check algorithm for group activityresponse analysis, and the Response-Crowd-Homogeneity-Check algorithm for group message esponse analysis. Anyalarm from either of these two algorithms will trigger abotnet alert/report.

SpyBot also has the ability to detect botnet C&C even when there is only one bot in the monitored network, if certain conditions are satisfied. This is discussed in Section 3.2.1 Response-Crowd-Density-Check Algorithm The ntuition behind this basic algorithm is as follows. For each time window, we check if there is a dense response crowd2. Recall that a group is a set of clients that connect to the same server. Within this group, we look for any essage or activity response behavior. If the fraction of clients with message/activity behavior within the group is arger than a threshold (e.g., 50%), then we say these responding clients form a dense response crowd.

3.2.2 Response-Crowd-Homogeneity-Check Algorithm The intuition of this algorithm is that, instead of looking at the density of response crowd, it is important to consider the homogeneity of a crowd. A homogeneous crowd means that within a crowd, most of the members have very similar responses. For example, the members of a homogeneous crowd have message response with similar structure and content, or they have scan activities with similar IP addressdistribution and port range. We note that we currently implement this algorithm only for message response nal- ysis.

4 Discussion and Future Work

4.1 Possible Evasions and Solutions

Evasion by encryption. Botnets may still use known protocols (IRC and HTTP) that SpyBot can recognize, but the otmasters can encrypt the communication content to attempt to evade detection. First of all, this may only mislead essage response correlation analysis, but can- not evade activity response correlation analysis. Second, we can mprove message response correlation analysis to deal with encrypted traffic. For example, instead of using simple ICE distance to calculate the similarity of two messages, we can use information-theoretic metrics that are relatively resilient to encryption, such as entropy, or normalized compression distance (NCD [4, 28]), which is based on olmogorov complexity. Evading protocol matcher. Although botnets tend to use existing common protocols to build their C&C, they may use some obscure protocols or even create their own protocols4. It is worth noting that "push" and "pull" are

the two representative C&C styles. Even when botnets use other protocols, the spatial-temporal correlation and similarity properties in "push" and "pull" will remain. Thus, our detection algorithms can still be used after new protocol matchers are added. We are developing a generic C&C-like protocol matcher that uses traffic features such as BPP (bytes per packet), BPS (bytes per second), and PPS (packet per second) [20, 26] instead of relying on protocol keywords. This protocol matching approach is based on the observation that there are generic patterns in botnet C&C traffic regardless of the protocol being used. For example, C&C traffic is typically low volume with a just a few packets in a session and a few bytes in a packet. Ultimately, to overcome the limitations of protocol matching and protocol-specific detection techniques, we are developing a next-generation botnet detection system that is independent of the protocol and network structure used for botnet C&C
.
Evasion by using very long response delay. A botmaster may command his bots to wait for a very long time (e.g., days or weeks) before performing message or malicious activity response. In order to detect such bots using Bot-Sniffer, we have to correlate IRC or HTTP connection records and activity events within a relatively long time window. In practice, we can perform correlation analysis using multiple time windows (e.g., one hour, one day, one eek, etc.). However, we believe that if bots are forced to use a very long response delay, the utility of the botnet to otmaster is reduced or limited because the botmaster can no longer command his bots promptly and reliably. For example, the bot infected machines may be powered off or disconnected from the Internet by the human sers/owners during the delay and become unavailable to the botmaster. We can also use the analysis of activity response crowd homogeneity to defeat this evasion. For example, if we can observe over a relatively long time window that several clients are sending spam messages with very similar contents, we may conclude that the clients are part of a botnets.

Evasion by injecting random noise packet, injecting ran- dom garbage in the packet, or using random response elay. Injecting random noise packet and/or random garbage in a packet may affect the analysis of message response rowd homogeneity. However, it is unlikely to affect the activity response crowd analysis as long as the bots still need to per-form the required tasks. Using random message/activity re- sponse delay may cause problems to the Response-Crowd- Density-Check algorithm because there may not be suffi- cient number of responses seen within a time window for one round of TRW. However, the botmaster may lose the reliability in controlling and coordinating the bots promptly if random response delay is used. We can use a larger time window to capture more responses. Similar to evasion by long response delay discussed above, for evasion by random response delay, a better solution is to use the analysis of activity response crowd homogeneity

In summary, although it is not perfect, SpyBot greatly enhances and complements the capabilities of existing bot-net detection approaches. Further research is needed to improve its effectiveness against the more advanced and evasive botnets.

5 Conclusion

Botnet detection is a relatively new and a very chal- lenging research area. In this paper, we presented Bot-Sniffer, a network anomaly based botnet detection system that explores the spatial-temporal correlation and similarity properties of botnet command and control activities. Our detection approach is based on the intuition that since bots of the same botnet run the same bot program, they are likely to respond to the botmaster's commands and conduct attack/fraudulent activities in a similar fashion. SpyBot employs several correlation and similarity analysis algorithms to examine network traffic, identifies the crowd of hosts that exhibit very strong synchronization/correlation in their responses/activities as bots of the same botnet. We reported experimental evaluation of SpyBot on many real- world network traces, and showed that it has very promising detection accuracy with very low false positive rate. Our ongoing work involves improving the detection accuracy of SpyBot and its resilience to evasion, and performing more evaluation and deploying SpyBot in the real-world. We are also developing a next-generation detection system that is independent of the protocol and network structure used for botnet C&C.

References

[1] Hi-performance protocol identification engine.
[2] Quick analysis of a proxy/zombie network.