A Detection Method for Botnet based on Behavior Features[1]

Weiming Li[1,a], Songlin Xie[2,b], Jie Luo[3,c], Xiaodong Zhu[4,d,*]

[1,2,3](School of Computer Science, Huazhong University of Science and Technology, 430074, China)

[4](hunan university of technology, 412000, china)

[a]lwm@hust.edu.cn, [b]107870814@qq.com, [c]rodger1017@163.com, [d]8458329@qq.com

Key words: network security; botnet; behaviors feature; similarity

Abstract: How to detect Botnet has become a very important problem in security network. The existent detection methods based on network traffic and host behaviors can't handle the emergency Botnets. In this paper we present an optimized method to analyze the similarity and time period of Botnets behaviors. In the end, our method gets an effective result. Our method uses the IDS-like architecture, which develops six specific components to detect six important Botnets abnormal behaviors. And it builds correlation rules to calculate match score. Through the experiments described in the paper, we can see that our method can not only detect already known Botnets precisely, but also detect unknown Botnets to some extent. The experiments prove that our method is effective and it has some advantages compared with other methods. At last, the paper proposes the future direction and the points that need to be improved.

Introduction

Botnet is the main threaten of Internet nowadays. A Botnet often controls thousands of zombie hosts, and it can launch all kinds of attacks including DDoS, worms spreading, spam mails etc. Moreover, a Botnet can lead to network security disasters. Thus, detecting Botnet is now an emergent request for network security. In this paper, we present a Botnet detection method based on behavior features that can find out most Botnets precisely.

There are two basic kinds of Botnet detection methods: one based on network flow features and the other based on behavior features. The former searches the keywords in Botnet command and control packets, and then gets the result through keywords matching. It's effective to detect IRC or HTTP Botnets, however, it can't analyze the encrypted network flow. The latter can detect all kinds of Botnets by analyzing abnormal behaviors of zombie hosts, but it's much more complex.

The method based on flow features is presented earlier. Strayer et al. [1] described their research of finding command and control channel through network bandwidth, connection timeout and packet size. Livadas et al. [2] applied machine learning technologies into detecting IRC Botnets. During their work, Bayes network was used to firstly classify IRC flow and non-IRC flow, then classify normal IRC channels and Botnet IRC channels. Binkley et al. [3] combined IRC abnormal features and TCP abnormal features to detect Botnets.

The method based on behavior features is now the main research direction. Karasaridis et al. of AT&T Lab [4] described a method to detect Botnet hosts in ISP backbone, which firstly fuse alerts and find suspect hosts, then use three heuristics rules to recognize Botnet hosts. Bothunter [5] is a famous project, using IDS-driven dialog correlation to detect Botnet hosts, which develops two snort plug-ins, SLADE and SCADE. SLADE implements connection n-gram payload analyze, finding out abnormal bytes distribution of Botnet packets. SCADE implements network scanning detection and searches Botnet hosts ingress and egress scanning. BotHunter correlator will relate all the alerts and form a detail Botnet infection dialog. BotSniffer [6] presents a network abnormality based on method detecting Botnet command and control server. The main idea is that connections between Botnet hosts and server have common features and time sequence similarity. BotMiner [7],

[1] Supported by Natural Science Foundation of Hubei Province(Grant No. 2011CDB038). Supported by the Creative Funds of HUST

* Corresponding Author

improving the BotSniffer, can detect distributed Botnets besides centralized Botnets. And it's more effective.

In this paper, we present a new method based on behavior features. And we also get a new approach to calculate similarity during the entire method. Our method can detect both known and unknown Botnets, its architecture is showed as Fig. 1.

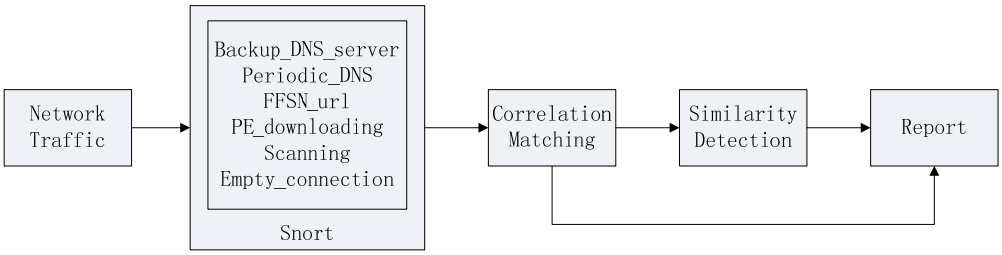

Figure 1. Architecture

Fig. 1 describes the core flow of the method and it is a NIDS-like Botnet detection system. In this paper, we add six plug-ins into Snort IDS to detect six different types of security events. The result alerts will be correlated by correlation rules which are designed to figure out Botnet behaviors. The final report is stored in database and can be sent to remote server.

Behavior Features Selection

In this paper, based on all kinds of known Botnets, we select six different behaviors as Botnets evidences. The six different behaviors are more effective and easier to collect than other methods.

Accessing Backup DNS Server. A normal host will send the request to main DNS server when it resolves a domain name. If access is failed, it will send the request to backup DNS server. However, Botnets often access both main and backup DNS servers at the same time to improve its availability. Thus, this abnormal accessing to backup DNS server is a kind of evidence.

Resolving Specific Domain Name Periodically. Once a Bot runs, it connects to command and control server for resolving the server's domain name. The Bot will send the resolving request constantly because Botnet command and control server is always unavailable (just temporal). As a result, the sending frequency is increasing dramatically. H.Choi [8] described the behavior feature.

Accessing Fast-Flux Service Network. One FFSN domain name maps to lots of IP addresses. And it's different from Content Distribution Network. Thorsten [9] demonstrated the differences. Through Emanuele [10]'s deep study into the Botnet FFSN behavior, the researchers find that Botnet master controls numerous zombie hosts and by adopting FFSN technology can avoid being tracked.

Downloading Malicious Binary Code. Botnet is essentially one kind of malicious code. It updates binary codes frequently to prevent from being cleaned by the anti-virus application. The malicious binary codes transformation during the network flow is an important behavior feature of Botnet.

Scanning. According to the scanning direction, Botnet scanning can be classified either as ingress scanning or egress scanning. Ingress scanning is that zombie host scans victim hosts. Egress scanning is that infected victim host scans other victim hosts. Both two steps are important features in Botnet life cycle.

Creating Null TCP Connection Periodically. Botnet hosts keep silence for a long time and wait for the commands from command and control channel. During hiding period, Botnet hosts keep contact with command and control server by null TCP connections. That is the most common way in all ends of Botnet. These null TCP connections also have a fixed frequency in time sequence. Generally, there are two kinds of null TCP connection, one is zero TCP windows, and the other is IRC PING/PONG connection.

In this paper, we define six types of Snort alerts on the basis of these six behavior feature.

Backup DNS Server. We use network flow sniffer to detect accessing of backup DNS server. When the destination address of TCP connection is the backup DNS server, the source host of the

Published by Atlantis Press, Paris, France.

corresponding TCP connection will be record as suspicious Botnet host.

Periodic DNS. We use Fast Fourier Transform to test a host's DNS requests for the same domain name. If Fast Fourier Transform finds some periods, the host will be added in suspicious Botnet hosts list.

FFSN URL. How to detect FFSN domain name is a complex problem. We use the method proposed in the research [11]. The core algorithm is to build an array of quad <A record number, A record dispersion, TTL, domain name creating time>, according to the DNS answers. If more than three items are over the threshold, we identify that the domain name is a FFSN domain name.

PE Downloading. We implement a plug-in of Snort, assembling network packets, searching PE header and malicious code features in a TCP stream. The malicious code features come from clamAV [11] official WEB site.

Scanning. We use Snort plug-in, SCADE, to detect TCP SYN scanning, TCP Connect scanning, UDP scanning and ICMP scanning.

Empty Connection. We firstly collect TCP flow information, and then use FAST Fourier Transform to find out the periods. If a TCP connection is an IRC connection containing PING/PONG command only, we identify that it is an IRC null connection.

According to the six types of alerts above, we can monitor a zombie host's almost entire life cycle as show in Fig. 2.

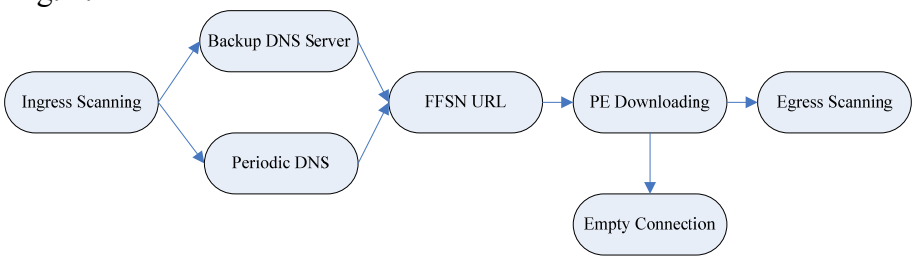

Figure 2. Monitoring Zombie Host's Life Cycle

The Correlation Rules

Correlation Rules Syntax. A correlation rule is consisted of "header", "content" and "Botnet name". The "header" defines the alert method. The "content" defines all kinds of behavior features. The "Botnet name" defines the rule related to the known Botnet name. For an instance:

alert backup_DNS_server: "xx.sqlteam.info"; periodoic_DNS: "xx.sqlteam.info"; scanning: scan_syn time_window:1500 msg: " Backdoor.Adverbot.A"
alert backup_DNS_server: "winudpmgr.mydyn.net"; FFSN_url: "winudpmgr.mydyn.net"; PE_downloading: "x80x90x80x90x80x90"; time_window:600 empty_connection msg: " Trojan.Ircbot.XI"

The first rule describes that:

- One host accesses to backup DNS server and the domain name is "xx.sqlteam.info";
- The host will periodically resolve the "xx.sqlteam.info" name;
- The host scans other hosts by TCP SYN packets;
- All the behaviors happen during 1500 seconds.

The second rule describes that:

- One host accesses to backup DNS server and the domain name is "winudpmgr.mydyn.net";
- FFSN name is "winudpmgr.mydyn.net";
- The host downloads a PE binary code containing "x80x90x80x90x80x90";
- All the behaviors happen during 600 seconds.
- Then the host keeps null TCP connection with command and control server.

Alerts Matrix. The core data structure of the method is an alerts matrix. The matrix stores current alerts and their detail information. Every alert is stored as a structure alert node.

Published by Atlantis Press, Paris, France.

```
typedef struct alertNode{
    string msg;
    timestamp timestamp;
    int order;
    long interval;
    alertNode *next;}
```

An alert node include 5 properties: “msg” is detail information; “order” is the alert indexes of the same host; “interval” is the period of time from the last alert to this one. If the same type alert appears multiple times, they are connected by the pointer “next”.

For an instance, an alert node represents a scanning behavior as follows.

```
scan_alertNode.msg = “scan_syn”;
scan_alertNode.timestamp = “2010-03-31 10:21:32”;
scan_alertNode.order = 5;
scan_alertNode.interval = 120;
scan_alertNode.next = null;
```

Fig. 3 is the alerts matrix which includes all currents alert nodes.

Host	Backup_DNS Server	Period_DNS	FFSN_url	PE_downloading	Scanning	Empty_connection
192.168.1.10	alertNode	alertNode		alertNode		alertNode
192.168.1.11	alertNode		alertNode		alertNode	
192.168.1.12	alertNode	alertNode		alertNode		
……						
192.168.1.236	alertNode		alertNode		alertNode	

Figure 3. The Alerts Matrix

How The Alerts Matrix Match Correlation Rules. Every line in the alerts matrix is the abnormal behaviors of one host. When an alert node is added in a line, the line will match all the correlate rules. If the match is correct, the host in this line is a zombie host affected by Botnet. A daemon will scan all the alert nodes and remove timeout alert nodes. There may be a situation that a line in the alerts matrix matches several correlation rules. Every match will be calculated at a score, and the biggest score represents the correct correlation rule and corresponding Botnet.

The whole match procedure is listed as below.

(1) Translate all the correlation rules into a linked list;
(2) Read all alerts and fill every alert in the alerts matrix;
(3) For every new alert node, match the line to all correlation rules and calculate the scores;
(4) The correlation rule with biggest score is the successful match. And the system reports the alert to administrator;
(5) Remove timeout alert nodes;
(6) Go to (3).

Experiments

Known Botnets Detection. We use many typical Botnets to test the effectiveness of the method which proposed above. These Botnets include IRC-Botnet, HTTP-Botnet and P2P-Botnet.

Table 2. Botnet Name and Abnormal Behaviors

Botnet	Backup DNS Server	Periodic DNS	FFSN	Scanning	PE Downloading	Null Connection
Exploit.DCOM.Gen	√	√	√			√
Exploit.MS04-011	√	√				
Generic.Sdbot.F6231A4E	√	√				
Trojan.SdBot-5053	√	√				
Zbot.sxq	√	√				
W32.Virut.bi			√		√	√
Trojan.Poebot	√	√	√			

Backdoor.Agobot.DL	√			√		
Generic.Sdbot.2D2F14CE		√			√	
Trojan.Mybot-5123	√	√				
Worm.Gaobot.102	√		√			
Worm.Gaobot.373	√	√		√		
Win32.Worm.Bobax-1			√		√	√
Packer.Krunchy.B	√			√	√	√
Trojan.Spybot.gen-3	√		√			√
Worm.P2P.KWBot.C	√	√				
W32.Virut.ca	√		√		√	√
Trojan.Poebot-29	√		√			√
Trojan.IRCbot-1944	√	√				
W32.Virut.ia	√		√		√	√

We define the correlation rules to detect these Botnets, and active all these Botnets in a network test bed. The Detection result is of well quality.

Table 3. Known Botnets Detection Result

Botnet	Infected hosts	Detected Hosts Number	Detection Rate	False Positive	False Negative
Exploit.DCOM.Gen	300	265	88.33%	0	11.67%
Exploit.MS04-011	300	267	89.00%	0	11.00%
Generic.Sdbot.F6231A4E	300	275	91.67%	0	8.33%
Trojan.SdBot-5053	300	242	80.67%	0	19.33%
Zbot.sxq	300	283	94.33%	0	5.67%
W32.Virut.bi	300	271	90.33%	0	9.67%
Trojan.Poebot	300	238	79.33%	0	20.67%
Backdoor.Agobot.DL	300	256	85.33%	0	14.67%
Generic.Sdbot.2D2F14CE	300	224	74.67%	0	25.33%
Trojan.Mybot-5123	300	249	83.00%	0	17.00%
Worm.Gaobot.102	300	252	84.00%	0	16.00%
Worm.Gaobot.373	300	230	76.67%	0	23.33%
Win32.Worm.Bobax-1	300	248	82.67%	0	17.33%
Packer.Krunchy.B	300	264	88.00%	0	12.00%
Trojan.Spybot.gen-3	300	275	91.67%	0	8.33%
Worm.P2P.KWBot.C	300	266	88.67%	0	11.33%
W32.Virut.ca	300	240	80.00%	0	20.00%
Trojan.Poebot-29	300	258	86.00%	0	14.00%
Trojan.IRCbot-1944	300	283	94.33%	0	5.67%
W32.Virut.ia	300	259	86.33%	0	13.67%

Unknown Botnets Detection. We use the predefined correlation rules to detect unknown Botnets, which means the abnormal behaviors of these Botnets are unknown to us, and there is no correlation rule corresponding to these Botnets. Although errors for Botnet name appear among the alert reports, the method still gets a good result. And the result shows that the method based on behavior features can be expended to detect new Botnets.

Table 4. Unknown Botnets Detection Result

Botnet	Infected	Detected Bots	Detection	False	False
Exploit.DCOM.Gen	300	212	70.67%	0	29.33%
Exploit.MS04-011	300	208	69.33%	0	30.67%
Generic.Sdbot.F6231A4E	300	169	56.33%	0	43.67%
Trojan.SdBot-5053	300	213	71.00%	0	29.00%
Zbot.sxq	300	220	73.33%	0	26.67%
W32.Virut.bi	300	176	58.67%	0	41.33%
Trojan.Poebot	300	205	68.33%	0	31.67%
Backdoor.Agobot.DL	300	189	63.00%	0	37.00%

Published by Atlantis Press, Paris, France.

Generic.Sdbot.2D2F14CE	300	183	61.00%	0	39.00%
Trojan.Mybot-5123	300	204	68.00%	0	32.00%
Worm.Gaobot.102	300	213	71.00%	0	29.00%
Worm.Gaobot.373	300	210	70.00%	0	30.00%
Win32.Worm.Bobax-1	300	176	58.67%	0	41.33%
Packer.Krunchy.B	300	184	61.33%	0	38.67%
Trojan.Spybot.gen-3	300	219	73.00%	0	27.00%
Worm.P2P.KWBot.C	300	185	61.67%	0	38.33%
W32.Virut.ca	300	163	54.33%	0	45.67%
Trojan.Poebot-29	300	190	63.33%	0	36.67%
Trojan.IRCbot-1944	300	212	70.67%	0	29.33%
W32.Virut.ia	300	186	62.00%	0	38.00%

Conclusion

In this paper we present a detection method to detect Botnets based on behavior features. It's capable of detecting both known and unknown Botnets. And it's an improvement in the research fields of Botnets detection. In the future, the method will be applied into testing for more Botnets and correlation rules.

References

[1] T. Strayer, D. Lapsley, R. Walsh, and C. Livadas, Botnet detection based on network behavior, Botnet Detection: Countering the Largest Security Threat, in Series: Advances in Information Security, Vol. 36, W. K. Lee, C. Wang, D. Dagon, (Eds.), Springer, 2008, pp1-24.

[2] C.Livadas, R. Walsh, D. Lapsley, T. Strayer, Using machine learning techniques to identify botnet traffic, in Proceedings 2006 31st IEEE Conference on Local Computer Networks, 2006, pp. 967-974.

[3] J.R. Binkley and S. Singh, An algorithm for anomaly-based botnet detection, USENIX SRUTI: 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet, 2006, pp43-48.

[4] A. Karasaridis, B. Rexroad, and D. Hoeflin, Wide-scale botnet detection and characterization, in Proceedings of the 1st Conference on 1st Workshop on Hot Topics in Understanding Botnets, Cambridge, MA, 2007.

[5] G.Gu, P.Porras, V.Yegneswaran, M.Fong, W.Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation, in 16th USENIX Security Symposium, 2007.

[6] G.F. Gu, J.J. Zhang, and W.K. Lee, BotSniffer: detecting botnet command and control channels in network traffic, in Proceedings of the 15th Annual Network and Distributed System Security Symposium, San Diego, CA, 2008.

[7] G.F. Gu, R. Perdisci, J.J. Zhang, and W.K. Lee. BotMiner: clustering analysis of network traffic for protocol- and structure-independent Botnet detection, in Proceedings of the 17th USENIX Security Symposium, San Jose, CA, 2008.

[8] H.Choi, H.Lee, H.Lee, and H.Kim, Botnet detection by monitoring group activities in DNS traffic, in Proceedings of the 7th IEEE International Conference on Computer and Information Technology, Washington D.C., USA,2007, pp.715-720.

[9] The Honeynet Project. Know Your Enemy: Fast-Flux Service Networks, 2007.

[10]T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In Proc. network and Distributed System Security Symposium, 2008.

[11]Information on http://www.clamav.net/lang/en/.

Published by Atlantis Press, Paris, France.