Defence Science Journal, Vol. 62, No. 3, May 2012, pp. 174-179, DOI: 10.14429/dsj.62.1425

Pattern Programmable Kernel Filter for Bot Detection

Kritika Govind*, Vivek Kumar Pandey, and S. Selvakumar
National Institute of Technology, Tiruchirappalli-620 015, India
**E-mail: kritika@nitt.edu*

ABSTRACT

Bots earn their unique name as they perform a wide variety of automated task. These tasks include stealing sensitive user information. Detection of bots using solutions such as behavioral correlation of flow records, group activity in DNS traffic, observing the periodic repeatability in communication, etc., lead to monitoring the network traffic and then classifying them as Bot or normal traffic. Other solutions for Bot detection include kernel level key stroke verification, system call initialization, IP black listing, etc. In the first two solutions there is no assurance that the packet carrying user information is prevented from being sent to the attacker and the latter suffers from the problem of IP spoofing. This motivated us to think of a solution that would filter out the malicious packets before being put onto the network. To come out with such a solution, a real time bot attack was generated with SpyEye Exploit kit and traffic characteristics were analyzed. The analysis revealed the existence of a unique repeated communication between the Zombie machine and the botmaster. This motivated us to propose, a Pattern Programmable Kernel Filter (PPKF) for filtering out the malicious packets generated by bots. PPKF was developed using the windows filtering platform (WFP) filter engine. PPKF was programmed to filter out the packets with unique pattern which were observed from the bot attack experiments. Further PPKF was found to completely suppress the flow of packets having the programmed uniqueness in them thus preventing the functioning of bots in terms of user information being sent to the Botmaster.

Keywords: Command and control, SpyEye exploit kit, WFP-windows filtering platform, kernel, zombie

1. INTRODUCTION

Bot is a malicious piece of software derived from the word 'robot', which when installed in a host makes it a Zombie machine. Botnets are a group of distributed bots controlled by a master computer often referred to as botmaster (remote attacker). A zombie machine may respond to genuine requests as well as the requests from its botmaster simultaneously. The responses for the genuine requests do not exhibit similarity. But the responses for the botmaster exhibit a high degree of similarity. This similarity can be used to distinguish the presence of malware. Many modern botnets rely on dynamic/fast-flux DNS services. In a fast-flux DNS technique, hundreds or thousands of compromised hosts are used as proxies to hide the identities of the true command and control (C&C) servers. These hosts constantly alternate in a round-robin fashion to resolve one hostname to many different IP addresses[1]. Bots can basically be classified into two types.

1.1 Network-based Bots

These bots generate attacks on the victim machine in an attempt to overload the computing resources. These bots perform various attacks such as spamming, phishing, clickjacking, and different flooding attacks, viz., UDP flood attacks, SYN flood attacks, Socket flood attacks, and HTTP flood attacks.

1.2 Host-based Bots

These are bots which get installed in the victim machine automatically when the user downloads any kind of freeware from the internet. These are Trojans which are bound along with the software downloaded from the internet using binder software which the victim is totally unaware of. These bots are developed with features such as keylogging for closely monitoring user behavior, interception of sensitive data including passwords and monitoring of mouse clicks. Bots usually communicate with their botmaster in order to receive commands from them. Based on this behavior, bots can be categorised into many types as discussed in the following

IRC bots make use of internet relay chat protocol for communication. The detection of such kinds of bots is done by using number of features such as average packet size, ratio of visible to invisible users in the channel, unusual or common nicknames used during the chat, average word length, string distance, etc.[2] Alternatively the command and control of botmaster can be implemented by message exchanges using HTTP. Such bots are known as HTTP bots. These HTTP bots are detected by observing the degree of periodic repeatability in bot communication using packet information such as source and destination IP addresses and the inter arrival time between the packets[3]. In a Peer2Peer(P2P) network, attackers make use of the decentralized structure of

Received 07 January 2012, revised 27 March 2012, published online 03 May 2012

the peer to peer protocol to implement P2P botnets, utilizing the availability of more bandwidth, storage, and computational power. A case study on Trojan. Peacomm bot shows the realization of P2P botnets, where the bots use Trojan for the primary injection and the P2P network for downloading the secondary injections[4]. Spambots are pieces of malware that are responsible for generation and delivery of spam mails. Spambots can be identified by examining the type, header, and contents of a single data packet[5]. Click bots are generated by clickjacking attack, where the victim is made to click on an element of a malicious webpage as created by the attacker. Vulnerabilities in JavaScript and HTML/CSS features are misused to create such attacks. These attacks can be avoided by taking the coordinates of clickable elements on a webpage[6]. Bots are known to generate DDoS attacks, wherein the attack is generated from multiple IP addresses in order to saturate a target network, using all the available bandwidth. Flooding attacks are usually prevented by rate limiting or black listing the IP addresses[7].

2. RELATED WORK

Bots are characterized based on their responses to remote control. This is done by observing the way bots and the genuine programs communicate over the network. Bots also create files and interact within the network similar to genuine programs. But these bots can be identified by distinguishing between the remotely initiated and locally initiated actions based on the system calls invoked[8]. Botnets are detected on observing the unique pattern of group activities among the botnet DNS traffic and genuine DNS traffic[9]. An anomaly based detection method exists, where the bots and their controllers are identified by summarizing the flow records between the local and the remote attacker machine on a particular port for over a considerable period of time[10]. In the detection of botnets based on their communication patterns, the packets filtered from the captured network traffic are monitored, clustered and are then sent through a flow analyzer and necessary reports are generated[11]. SLINGbot: A system for live investigation of next generation botnets characterizes bots based on the command and control traffic in a simulated environment[12]. BlackEnergy is a web-based bot capable of generating distributed denial of service (DDoS) attacks. These bots aim at bombarding the victim machine with DDoS attacks and use a runtime encrypter for preventing its detection by the anti-virus. Based on the analysis it is seen that these bots use HTTP to communicate to the C&C server by sending a POST message to the server[13].

3. MOTIVATION

It was found from literatures[2,9] that bot exhibits uniqueness in maintaining connectivity with their botmasters through a regular exchange of messages. This led us to use this communication pattern as a key component for bot detection. Hence in this paper, a kernel level packet filtering module namely, pattern programmable kernel filter (PPKF), which would filter out the malicious packets generated by bots, based on their repetitive communication pattern is proposed.

4. PROPOSED WORK

4.1 Pattern Programmable Kernel Filter

The block schematic of the pattern programmable kernel filter (PPKF) as shown in Fig.1 gives the basic flow of the filter wherein the outbound packets are captured by the kernel packet capture module. Then the pattern inference module (PI) analyzes the packets and identifies the pattern in them. If found, then the unique pattern is programmed into the PPKF module thus enabling it to filter out all the malicious packets at kernel level before being put onto the network. Fig. 2 shows the positioning of components of the proposed system into the kernel space. Notations A_1, A_2, …, A_n, indicate the existence of various applications in the user space, which may generate genuine as well as malicious packets. Packets generated by these applications are captured and are processed by the PI module for inference of pattern, if any. If found then the PPKF is programmed with this pattern to filter out the malicious packets generated by bot. According to the proposed system the pattern can either be generated automatically at kernel level from the packet payload or it can be inferred from network traces through offline analysis and then the PPKF can be programmed accordingly. Pattern for SpyEye bot was obtained in this paper from offline analysis from the network traces and then programmed into the PPKF.

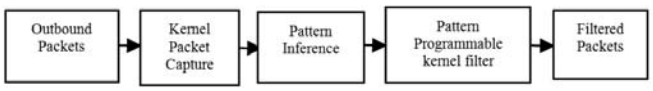

Figure 1. Block schematic of pattern programmable kernel filter.

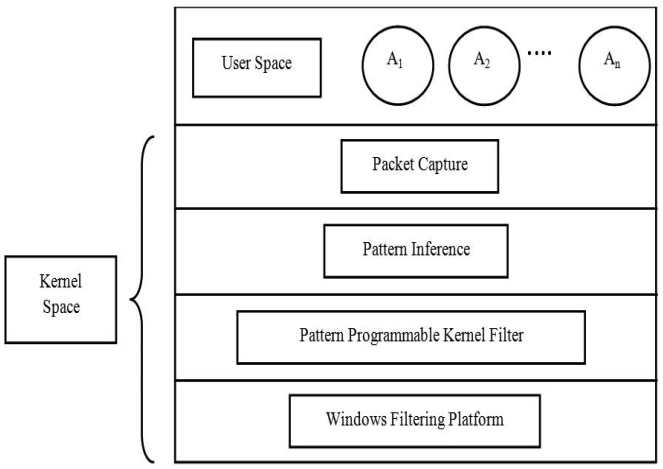

Figure 2. Positioning of the proposed PPKF into kernel space.

5. WORK DONE

The effective working of the proposed system was tested by generating a bot attack in and using the PPKF driver for its mitigation. Bot attack was generated with the recently released SpyEye exploit kit. Traffic analysis was done using wireshark and malicious traffic pattern with respect to SpyEye bot was identified.

5.1 Experimental Setup

SpyEye Tool Kit: The important feature of this exploit kit is to steal user information such as username, password, and other sensitive data of the user from the victim machine. This is a web based tool kit with command and control facilities. The kit basically builds a backdoor Trojan which is sent to the victim machine to take control of the PC. During the bot building process the tool provides the following options: Path to the main control panel, time interval, UPX, and Kill Zeus option. The path to the main control panel specifies the path where the web server, with which the bot has to communicate, is set up. Time interval refers to the connector interval of time the bot has to connect to its master. UPX is the ultimate packer for executables, wherein the tool compresses all the executables into a single build, which is the malicious file to be sent to the victim machine. Running a SpyEye bot requires a web server running in a MySQL, PHP environment. The operating systems which support SpyEye range from windows 2000 to windows 7. SpyEye uses the hook function Wininet API, which is a Microsoft programming interface for windows that provides access to various protocols such as HTTP, FTP, and Gopher. The malicious bot program makes few registry changes during its first installation and thereby starts functioning in the victim machine[14,15].

5.2 Bot Attack Generation

An Intel Core 2 Duo processor machine with windows 7 (victim machine 1) OS installed in it was made the victim machine. The malicious build created using the SpyEye toolkit was sent to the victim machine as an E-mail attachment embedded along with a genuine picture file. Once the victim extracts the rar file to view the genuine picture, the bot gets installed on the victim machine thus making it a Zombie machine. The attacker machine which is also the C&C server was configured with WAMP to run the Apache web server with PHP, MySQL environment. The same malicious build file was sent to two more machines in the same LAN and their communication was noted. The successful attack generation is evident from Fig. 3 which are marked in boxes.

5.3 Result Analysis

The SpyEye bot used HTTP for communication. When the traffic was sorted with various filter expressions in wireshark, a regular exchange of HTTP packets between the Zombie machine and the Botmaster with a particular periodicity was observed. From the captured traffic, HTTP packets were filtered and graphs were drawn which showed all the nodes involved in communication with the zombie machine with their respective source and destination IP addresses. Further from the HTTP flow graph drawn as in Fig. 3 it is evident that the zombie machine is sending its own details, viz., computer name, username of the computer, status of the machine (online), version of OS, etc., to the attacker machine periodically. Thus based upon this communication, a pattern to be used for the detection of bots was identified and a pattern programmable Kernel level packet filtering algorithm is proposed for the mitigation of bots.

5.4 Pattern Programmable Packet Filtering Algorithm

From the Fig. 3, it is evident that the victim machine is reporting to the attacker machine constantly at various time instances say t1, t2, t3, t4. From this, the ΔT values viz., T1, T2, T3 are calculated which clearly show the

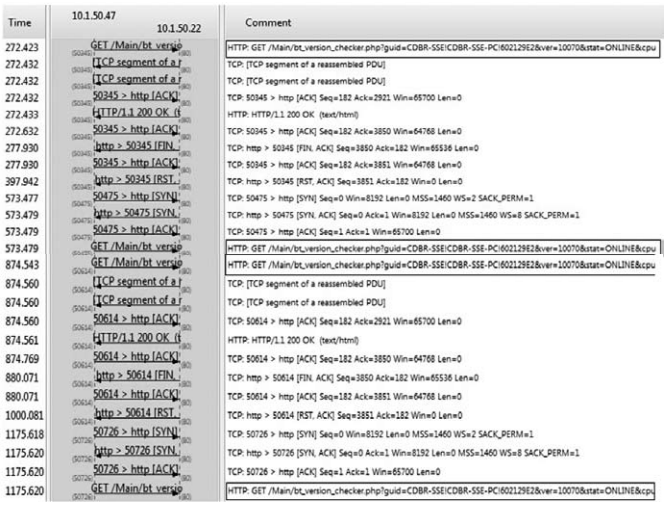

Figure 3. HTTP flow graph.

constant time interval between them, where t1, t2, t3, t4 are the arrival time of the packets in IST format.

t1 = 12:16:42 T1=t2-t1=5 mins
t2 = 12:21:43 T2=t3-t2=5 mins
t3 = 12:26:44 T3=t4-t3=5 mins
t4 = 12:31:45

Proposed PPKF Algorithm

Step 1: Capture traffic at kernel level
Step 2: Filter outbound TCP packets
Step 3: Inspect
If (packets = GET/POST packets)
Then Pattern Check;
If Pattern Found
a. Compute Periodicity
b. Perform Remote Address Checking
If ((Periodicity Exists) && (Remote Address Check is verified) ||
(Periodicity Exists) && (Remote Address Check is not verified) ||
(Periodicity does not Exist) && (Remote Address Check is verified))

Then Block
Else Allow
Else Allow
Else Allow
Step 4: *End*
Pattern Check
If (TcpData==SpecifiedPattern)
Then Set Pattern Found = *True*
Else Set Pattern Found = *False*
Return

The proposed algorithm uses two levels of verification for bot detection namely, periodicity calculation, and remote address verification. In case of dynamic fast flux botnets, the pattern from the suspected packets may be the same along with periodicity in them but their IP addresses may differ. So the remote addresses are cross checked to look for similarities among them. Periodicity is also taken into account since the zombie machines report to their botmasters the user information periodically. Finding the exact pattern becomes the key component to filter out the malicious bot packets. Malicious packets in this context refer to the packets which the Botmaster sends to the victim to maintain its connectivity and also the packets sent from the victim machine in response to the packets from its botmaster.

5.5 Detection Module at Victim Machine for Kernel Level Packet Filtering

The appropriate signatures for blocking the malicious packets with respect to SpyEye bot were found to be HTTP packets containing the following keywords, 'Guid', 'Online', in their packet payload. Their communication pattern is discussed in part C of this Section. In similar fashion, signature for other bots can also be found and embedded into the PPKF. Further the bot attack was generated in three different environments, viz., a machine running windows XP, a machine running windows 7, and also on a virtual machine running windows 7. The results with respect to the communication pattern and periodicity were found to be the same in all the three cases. The developed PPKF was installed on the victim machine 1. A GUI called open system resource (OSR) was used to load/unload the packet filter in the victim machine. The loading/unloading of a filter can also be done manually which involves making registry changes manually and then starting the filter through command prompt.

Windows Filtering Platform: The PPKF was developed using the WFP filter engine. It is an application programming interface (API) that allows processing/filtering of packets in the network stack. It provides a filtering engine which can be used for packet filtering at kernel level. It is intended for use by various applications such as firewalls, and also for packet-processing. The WFP includes a filter engine, base filtering engine, and a callout. The filter engine provides the basic filtering capabilities which span across both the kernel and the user-mode. The Base filtering engine manages the filter engine. Callout functions are used for registering the filter rules and are invoked when the filtering conditions are matched[16].

6. PERFORMANCE ANALYSIS

Bot attack was generated in two different machines under three different scenarios and the communication behaviors of the victim machines were noted. Web traffic was generated by using three different web browsers, viz., Mozilla Firefox, Internet Explorer, and Google Chrome and the total number of packets between the Botmaster and zombie machine during the attack period were noted as shown in Table 1.

Table 1. Bot attack results in victim machine

Web browser used	Total no. of packets	No. of packets between BM and victim machine	No. of HTTP packets between BM and victim machine
Internet Explore	9951	72	11
Mozilla Firefox	13584	224	41
Google Chrome	17518	84	14

The packet capture results shown in the Table 1 correspond to the bot attack generated on victim machine 1. The results are based on the network traffic captured on the victim machine using wireshark. While using Internet Explorer and Mozilla Firefox as web browsers it was evident that the zombie machine effectively posted all user information such as web page browsed by the user, user name and password submitted in the web forms to the botmaster. But when Google chrome was used as web browser it was noticed that the HTTP 'Post' packets, viz., the packets containing information such as web page browsed by the user, user name and password were not sent to the botmaster. Instead only the HTTP reporting packets from the zombie machine to the Bot master were noticed, thus maintaining the Bot communication. So it is inferred that the botmaster still maintains connectivity with the zombie machine. This situation is averted by using the 'PPKF' wherein the filters all the malicious packets between the botmaster and the zombie machine thus totally suppressing the bot traffic. The results shown in Table 1 were with respect to the experiments tested on windows 7 machine.

The performance analysis of the proposed PPKF was done by measuring the number of packets in the victim machine before and after loading the filter. The captured network traffic from Figs. 4 and 5, show the total number of packets measured between the botmaster and zombie machine before and after installing the filter respectively. The results are tabulated in Table 2. The data shown in Table 2 has correlation along vertical columns 4, 6, 8, and 10 only. It is evident from the 2nd row of Table 2 that packets going to the botmaster are totally being suppressed by PPKF. Since the packet carrying user information from the bot infected machine is filtered

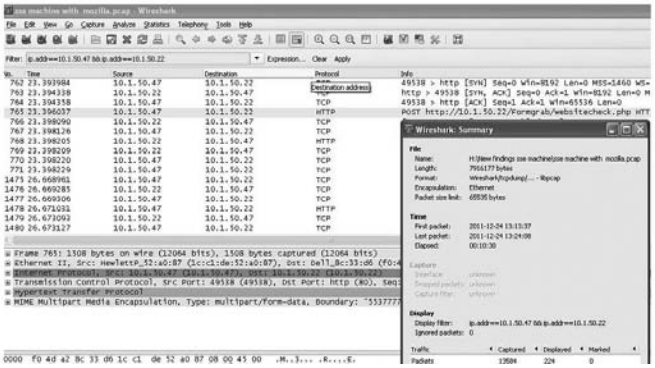

Figure 4. Total No. of packets between the Botmaster and zombie machine using Mozilla Firefox-before installing the PPKF driver.

out as seen from row 2, columns 4 and 6, there is no scope for the botmaster to maintain connectivity with the zombie machine which is evident from row 2, columns 8 and 10. This enables us to attain 100 per cent accuracy in terms of stopping the bot communication.

Figure 5. Total number of packets between the Botmaster and zombie machine-after installing the PPKF driver.

Table 2. Performance analysis of packet capture statistic in victim machine (Mozilla Firefox).

Total no. of packets		Outbound (From bot machine)				Inbound (To bot machine)			
		Total no.	To BM only	HTTP packets		Total no.	From BM only	HTTP packets	
				Total no.	To BM only			Total no.	From BM only
Before	13584	5484	129	662	19	6773	95	638	22
After	12898	3528	-	521	-	3903	-	516	-

7. CONCLUSION

Bots have become a major threat to the internet users of today given the rate of increased usage of Internet. The focus of this paper is to come out with an effective solution for the detection of Bot. Accordingly, a PPKF was proposed and implemented. Since SpyEye Exploit kit was recently released and not much research solutions were found addressing it, we used them for our experimental purpose. From the experiments it is evident that the proposed PPKF detects the presence of bot and suppresses the bot communication.

Our ongoing research work is on the automatic pattern inference module of PPKF (viz., to make the filter itself capable of finding malicious packet pattern) which will address the generic solution of filtering out malicious packet patterns generated by different bots. This feature of PPKF will also be effective in filtering out the malicious packets from other malicious sources such as worms which exhibit uniqueness in their communication pattern.

8. ACKNOWLEDGEMENTS

The authors thank the National Technical Research Organization (NTRO), New Delhi, Government of India for sponsoring this research work under the collaborative directed basic research in smart and secure environment project. Also we thank the anonymous reviewers for their critical review which improved the quality of this paper.

REFERENCES

1. Website referred for fast flux definition. http://my.safaribooksonline.com/book/-/9781597495356/2dot-botnet-overview/196#X2lud ODE1OTc0OTUzNTYvMTk2 [Accessed on 13 March 2011]
2. Mazzariello, Claudio & Sansone, Carlo. Anomaly-based detection of IRC botnets by means of one-class support vector classifiers. *In* Proceedings of the 15th International Conference Image Analysis and Processing - ICIAP 2009, Vietri sul Mare, Italy, September 2009. LNCS 5716, pp. 883-92.
3. Lee, Jae-Seo; Jeong, HyunCheol; Park, Jun-Hyung; Kim, Minsoo & Noh, Bong-Nam. The activity analysis of malicious HTTP-based botnets using degree of periodic repeatability. *In* Proceedings of International Conference on Security Technology, SECTECH'08, December 2008, Hainan Island, China. pp. 83-86.
4. Al , Yousof & Aickelin, Uwe. Behavioral correlation

for detecting P2P bots. *In* Proceedings of the Second International Conference on Future Networks, ICFN 2010, Sanya, Hainan, China, January 2010. pp. 323-327.
5. Li, Zhitang; Hu, Jun; Hu, Zhengbing; Wang, Bingbing; Tang, Liang & Yi, Xin Measuring the botnet using the second character of bots. *Journal of Networks*, 2010, **5**(1), 98-105.
6. Balduzzi, Marco; Egele, Manuel; Kirda, Engin; Balzarotti, Davide & Kruegel, Christopher. A solution for the automated detection of clickjacking attacks. *In* Proceedings of ASIACCS'10, Beijing, China, April 2010. pp. 135-44.
7. Williams, Craig. Exploring a Java bot: Pt 1. Cisco Blog, December 2009.http://blogs.cisco.com/security/exploring_a_java_bot_part_1/ [Accessed on 04 March 2010]
8. Stinson, Elizabeth; John, C. & Mitchell. Characterizing Bot's remote control behavior. *In* Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA '07, Lucerne, Switzerland, July 2007. pp. 89-108.
9. Choi, Hyunsang; Lee, Hanwoo; Lee, Heejo & Kim, Hyogon. Bot detection by monitoring group activities in DNS traffic. *In* Proceedings of the 7th IEEE International Conference on Computer and Information Technology, CIT '07, University of Aizu, Fukushima Japan, October 2007. pp. 715-20.
10. Karasaridis, Anestis; Rexroad, Brian & Hoeflin, David. Wide-scale botnet detection and characterization. *In* Proceedings of the 2nd conference on USENIX'08, Annual Technical Conference, Boston, Massachusetts, June, 2008. pp. 7-7.
11. Zeidanloo, Hossein Rouhani & Manaf, Azizah Bt Abdul. Botnet detection by monitoring similar communication patterns. *Int. J. Comp. Sci. Inf. Security*, 2010, **7**(3), 36-45.
12. Jackson, Alden W.; Lapsley, David; Jones, Christine; Zatko, Mudge; Golubitsky, Chaos & Strayer, W. Timothy. SLINGbot: A system for live investigation of next generation botnets. *In* Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security CATCH '09, Washington, DC, USA, March 2009. pp. 313-318.
13. Nazario, Jose. BlackEnergy DDoS bot analysis. Arbor Networks Security Blog, October 2007. http://ddos.arbornetworks.com/2007/10/blackenergy-ddos-bot-analysis-available/ [Accessed on 20 April 2010]
14. Mieres, Jorge. SpyEye Bot Analysis of a new alternative scenario crimeware. Malware Intelligence. February 2010.http://www.malwareint.com/docs/spyeye-analysis-en.pdf [Accessed on 09 November 2010]
15. Coogan, Peter SpyEye Bot versus Zeus Bot. Symantec blog. February 2010. http://www.symantec.com/connect/blogs/spyeye-bot-versus-zeus-bot [Accessed on 20 September 2010]
16. Website referred for the design of WFP filter driver http://msdn.microsoft.com/en-us/library/ff571068.aspx [Accessed on 10 January 2011]

Contributors

Ms Kritika Govind received her BE (Computer Sci. Engg) from Sakthi Marriaman Engineering College, Anna University, Chennai, Tamil Nadu, in 2009. She is working as a Research Assistant at Department of Computer Science and Engineering, National Institute Technology (NIT), Tiruchirappalli, Tamil Nadu. She is also pursuing her Master of Science (MS by Research) in Computer Science and Engineering at NIT, Tiruchirappalli. Her areas of interest include: Cyber security and network security.

Mr Vivek Kumar Pandey is pursuing his BE (Computer Sci. Engg) at National Institute of Technology, Tiruchirappalli. He has a keen interest towards coding and his field of interest includes: Network Security besides astrophysics.

Dr S. Selvakumar is a Professor in the Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India. He received his PhD from the Indian Institute of Technology Madras (IITM), Chennai in 1999. His research interests include group communication in high-speed networks, routing, multimedia communication, scheduling for QoS guarantee, mobile networks, network security, wireless sensor networks, and network computing. He has to his credit of publishing 54 research papers. He is currently the Investigator of the Collaborative Directed Basic Research–Smart and Secure Environment (CDBR-SSE) Project sponsored by NTRO, Government of India, New Delhi. He is presently the member of All India Board of IT Education, AICTE, New Delhi.