Next Generation Security Framework to Detect Botnets on Computer Networks

Asanka Balasooriya and Shantha Fernando

***Abstract*—Botnet is one of the most widespread and serious modern malware occurs commonly in today's cyber attacks. A botnet is a group of compromised computers which are remotely controlled by hackers to launch various network attacks, such as DDoS attack, spam, click fraud, identity theft and information phishing. The effort of the research is to analyze the behavior, possible countermeasures and preventive procedures of botnets; and come up with a next generation security framework to detect botnets on computer networks.**

***Index Terms*—Botnet, bots, centralized, decentralized, peer-to-peer, similar behavior.**

I. Introduction

The threat landscape has changed over recent times. It is no longer the teenagers who are trying to break into the systems but well organized criminals stealing sensitive information to make money. Large scale attacks and digital criminal activities have exposed the Internet to serious security breaches, and alarmed the world regarding cyber-crime. The cores of this problem are the so called botnets. A better understanding of Botnets will help to coordinate and develop new technologies to counter this serious security threat.

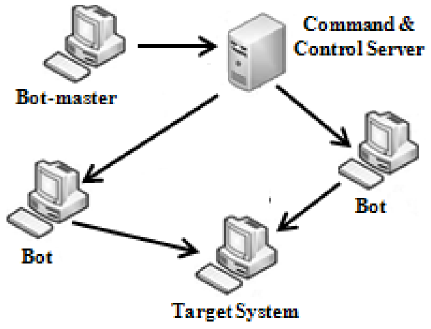


Fig. 1. An overview of basic Botnet functionality

II. Basics of Botnets

Botnets exist in many different forms [1]. Fig. 1 shows a basic overview of their basic functionality. A botnet is shown using a centralized architecture for it Command & Control channel, and engaging in worm-like propagation and a Distributed Denial of Service attack.

Botnets usually commandeer new victims by remotely exploiting a vulnerability of the software running on the victim. Botnets borrow infection strategies from several classes of malware, including self-replicating worms, e-mail viruses, etc. Fig. 2 shows various stages in a typical botnet life-cycle.

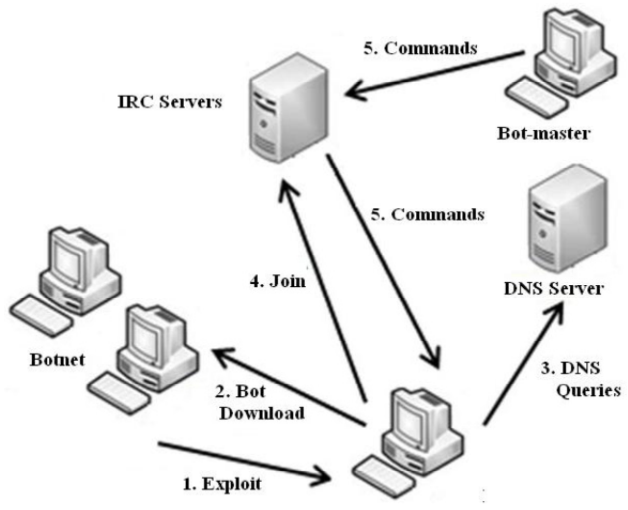


Fig. 2. A typical Botnet life-cycle.

III. Command and Control

Since the defining feature of a botnet is the ability of the bot-master to control the bots, some channel of communication must be present. This if often done by adding a simple protocol layer to an well-known protocol like HTTP or IRC, but sometimes more complex protocols functioning at lower levels in the network stack can be seen.

There are mainly two different Botnet Architectures as described in [1]. Those are, Client/server (centralized) botnets and Peer-to-peer (decentralized) botnets.

A. Client/Server Botnets

Historically, the type of Command & Control protocol encountered most often by researchers has been based on the Internet Relay Chat (IRC) protocol and IRC botnets are still in widespread use [2]. It also called as IRC, HTTP-based Centralized Command and Control (C&C) Botnet Architecture. This is the most common type of Botnet available in the internet.

The paper [3] concludes that 60% of the Botnets they found were IRC based Botnets and only a handful used HTTP for the C&C. Out of the C&C Botnets 70% of the botnets were single IRC base Botnets. Fig. 1 shows a typical Client/Server Botnet Architecture. The problem the bot-master faces using Client/Server architecture for the Command & Control channel, is presence of a central point of failure.

Manuscript received December 19, 2012; revised March 12, 2013.

The authors are with the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.(e-mail: asankabalasooriy@gmail.com, shantha@uom.lk)

B. Peer-to-Peer Botnets

In a peer-to-peer architecture, there is no centralized point for command and control. Nodes in a peer-to-peer network act as both clients and servers such that there is no centralized coordination point that can be incapacitated. If nodes in the network are taken offline, the gaps in the network are closed and the network continues to operate under the control of the attacker. Due to its peer architecture, it is very hard to detect Peer-to-Peer Botnets.

IV. RELATED WORK

Botnet detection is a very challenging problem and many researchers done research in this area. There are various methods to detect Botnets. There are several researches done by setting up a Honeynet which is integrated with Intrusion Detection Systems (IDS) to detect Botnets.

The concept of Honeypots in general is to catch malicious network activity with a prepared machine. This computer is used as bait. The intruder is intended to detect the Honeypot and try to break into it. Next, the type and purpose of the Honeypot specifies what the attacker will be able to perform. Often Honeypots are used in conjunction with Intrusion Detection Systems. There are many papers discussed how to apply honeynets for Botnet detection [4]-[8].

Signature based detection [9] is nothing but pattern matching. This extracts the features from the IRC packet and performs the cross check with the existing IRC C&C signatures stored in the database. If a match is found then it is declared as attack. The process of this method is easy because this compares simple byte sequences only. Moreover this kind of detection produces less or no false detections.

The interaction pattern or behavior of bots varies from human [10]. Human interaction occurs frequently and with varying Intervals. If the log of the bot traffic is examined, bots stay idle for a long time; once it receives the command from the Bot-master, it responds quickly and then stays idle until it receives the next command. Therefore the C&C channel detection becomes feasible by using spatial-temporal reasoning [11]. The inter arrival time between the C&C instructions of one bot will not vary or vary marginally when compared against another.

Flow characteristics like packets per flow (ppf) and average, bytes per packet (bpp), bytes per second (bps), packets per second (pps) have been used in separating the botnet traffic from the TCP traffic. These parameters can only help in separating aggressive flow. Modern attackers keep the rate as low as possible to masquerade the attack flow as normal. Hence to separate the low rate attack flow, the flow per IP address is correlated to find out the similar behavior among the flows.

BotSniffer [12] which is a network-based anomaly detection that identify botnet command and control channels without prior knowledge of signatures. It detects bots by examining the correlation and similarity patterns between bots activities within similar time window such as coordinated communication, propagation, attack and fraudulent activities due to the pre-programmed response activities to Bot-master commands.

For detecting Botnets Machine learning techniques are also used. Machine learning algorithms do not need explicit signatures to classify malware programs but rather is based on finding common features and correlating different activities of the malware. The papers [13] and [14] present machine learning techniques for botnet detection by using network statistics.

Detecting and neutralizing peer -to-peer based Command & Control channels is a more complicated task. There are two main solutions based on the study of Strom which are highlighted in research papers. The first solution is to pollute the Command & Control channel with false orders. That solution is only applicable on botnets using an unauthenticated publish/subscribe Peer-to-Peer architecture. The other solution proposed is an eclipse attack [15]. An eclipse attack attempts to divide the network into smaller networks by infiltrating the network with a large number of nodes, and preventing communication across these infiltrating nodes. The eclipse attack could be mounted as second stage of a Sybil attack [16].

V. NEXT GENERATION SECURITY FRAMEWORK

The next generation security framework is a security model to detect botnets on computer networks. This security model outlines how security is to be implemented on computer networks to detect botnets effectively. To analysis the botnets in computer network, freely available bot detecting tools and honey-pots will be deployed. This research will be useful for, security researchers to have a look at a new model to detect botnet and everyone who is interest on security to have an in-depth knowledge on Botnets.

The proposed Next Generation Security framework is based on passively monitoring network traffics. This model is based on the concept that multiple bots within the same Botnet will perform similar communication patterns and malicious activities. Fig. 3 shows the architecture of the proposed Next Generation Security Framework to Detects Botnet on Computer Networks. It consists of 6 main components. Those are Perimeter Filtering, Traffic Classifier, HTTP Based Bot Detector, IRC Based Bot Detector, SMTP Traffic Analyzer and Peer-to-Peer Bot Detector.

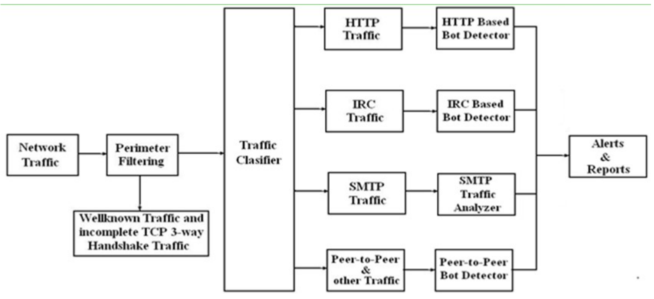


Fig. 3. Next Generation security framework

A. Perimeter Filtering

The main objective of Perimeter Filtering is to reduce the huge unwanted traffic workload and makes the rest of the system perform more efficiently. Following diagram shows the architecture of the filtering. In Transport connection Protocol (TCP) to establish a connection, uses a three-way handshake mechanism. In this Perimeter Filter, it will filter-out the traffics that the TCP handshaking have not completed. Like a host sends SYN packets without completing the TCP handshake. Most of these traffics are generated due to the scanning activities.

This Perimeter Filter not only filter-out the above mentioned incomplete TCP handshake packets but also filters out the huge well-known legitimate traffic. For that; need to gather the information about servers and their services provided to outside world. Then it is easy to filter out that well-known legitimate traffic from the Perimeter Filter.

B. *Traffic Classifier*

Filtered traffic from the Perimeter Filter piped into the Traffic Classifier. Traffic Classifier is responsible to separate IRC, HTTP and SMTP traffics from the rest of traffics and send them to corresponding Bot Detectors for analyze the traffic for Bots. For this to happened Traffic Classifier use standard TCP port numbers defined in the RFC6335[17]. Following is the summary of most commonly used TCP protocols and their associate port numbers.

For identifying HTTP traffic which are not using the default port 80 traffic , it is necessary to inspect the first few bytes of HTTP request and if it has certain pattern or strings, separate it and send it to HTTP Bot Detector. For detecting HTTP traffics we focus on concept of HTTP [18] protocol. Similar to most of other network protocols, HTTP uses the client-server model [19].

In HTTP, there are 3 methods. Those are “GET”, “HEAD”, or “POST”. So, it is necessary to inspect the first few bytes of an HTTP request contain “GET”, “POST” or “HEAP”. Then it is much more accurate to classify as HTTP and can send them to the HTTP Bot Detector.

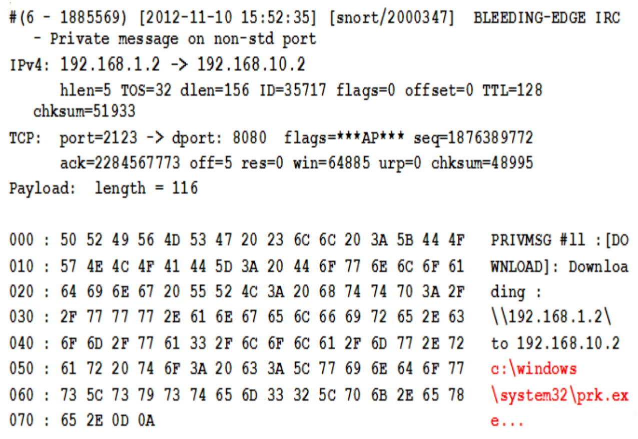
To detect IRC traffics those are not using standard port, it is necessary to inspect the contents of each packet and try to match the data against a set of user defined strings. By inspecting the first few bytes of the payload and looking for specific strings; it is much easier to catch IRC traffic. These IRC specific strings are NICK for the client’s nickname, PASS for a password, USER for the username and JOIN for joining a channel.

TCP Port 25 SMTP traffic is forward to the SMTP Traffic Analyzer for Detecting SMTP related bot activities. And rest of the traffic which are Peer-to-Peer and other traffic are forward to the Peer-to-Peer Bot Detector for detecting Bots.

C. *IRC Based Bot Detector*

The IRC [20] protocol regulates the recommended commands that should be used, for example NICK, JOIN, USER or MODE. USER names will in nearly all cases also be the same randomized NICK used to join the IRC. Some variations do occur, but spotting unusual NICK’s is the key to a successful bot detection program.

There are many online games that use IRC communications for game chat between the users. To identify accurately a Botnet from a game connection is to look at the number of alerts generated by the host/destination. If you notice a relatively few connection alerts, it’s most likely a Botnet. If you have hundreds or thousands of alerts in a short period, it’s in all likelihood a game or regular chat. We can verify this by examining the PRIVMSG alerts. Fig. 4 shows why it is important to catch and read the PRIVMSG’s when a bot detection in process.



```
#(6 - 1885569) [2012-11-10 15:52:35] [snort/2000347]  BLEEDING-EDGE IRC
   - Private message on non-std port
IPv4: 192.168.1.2 -> 192.168.10.2
      hlen=5 TOS=32 dlen=156 ID=35717 flags=0 offset=0 TTL=128
   chksum=51933
TCP:  port=2123 -> dport: 8080  flags=***AP*** seq=1876389772
      ack=2284567773 off=5 res=0 win=64885 urp=0 chksum=48995
Payload:  length = 116

000 : 50 52 49 56 4D 53 47 20 23 6C 6C 20 3A 5B 44 4F   PRIVMSG #ll :[DO
010 : 57 4E 4C 4F 41 44 5D 3A 20 44 6F 77 6E 6C 6F 61   WNLOAD]: Downloa
020 : 64 69 6E 67 20 55 52 4C 3A 20 68 74 74 70 3A 2F   ding :
030 : 2F 77 77 77 2E 61 6E 67 65 6C 66 69 72 65 2E 63   \\192.168.1.2\
040 : 6F 6D 2F 77 61 33 2F 6C 6F 6C 61 2F 6D 77 2E 72   to 192.168.10.2
050 : 61 72 20 74 6F 3A 20 63 3A 5C 77 69 6E 64 6F 77   c:\windows
060 : 73 5C 73 79 73 74 65 6D 33 32 5C 70 6B 2E 65 78   \system32\prk.ex
070 : 65 2E 0D 0A                                       e...
```

Fig. 4. PRIVMSG alert in snort IDS.

The PRIVMSG alerts are going to contain private messages from Internet Relay Chat conversations. There is a simple method to greatly reduce exposure of private messages that are not Botnet related. Namely, only look at private message alerts that have first matched up to a clearly random NICK or USER name. Then search on all alerts that match the IP addresses used by the IRC server and the suspected station. Then examine the alerts.

D. *HTTP Based Bot Detector*

The HTTP protocol is used in place of the IRC protocol and also port 80 is used. Because of the wide range of services used, it is not easy to block the central Command and Control server. By using the HTTP protocol, bots hide their communication flows among the normal HTTP flows, and avoid detection by the network defenders such as the firewalls [21]. Following is a HTTP GET request in a Botnet environment.

HTTP GET master/bb.php?id=5737x7x7x7x&v=300&tm =210&b=x11test

We can develop an IDS signature by looking for strings within the URI portion of the HTTP GET request. The analyst would focus on strings that appear to be part of the protocol employed by the C&C infrastructure, but that probably will not occur in normal HTTP traffic. In the above case, the rule could look for the presence of the strings .php? and 7x7x7x7x.

In our HTTP Based Bot Detector we implement an IDS as the primary detector to detect HTTP based Botnets according to signature availability and as the secondary method to detect rest of the HTTP based Botnet activities where there are no signatures available in the IDS we use the repeated HTTP GET request as polling method which is presented in the paper [22].

E. *Peer-to-Peer Bot Detector*

Peer-to-Peer Botnet has a decentralized command and control architecture. Modern Botnets highly use structured overlay topologies [23]. Modern botnets, such as Storm, Peacomm, and Conficker use these type of structured overlay networks [24]. Due to their lack of centralization, a botnet herder can join and control at any place. So, it is very hard to detect. Further, structured overlay mechanisms are designed to remain robust in the face of churn [25], an important concern for botnets, where individual machines may be frequently disinfected or simply turned off for the night. Not only have that but also structured overlay networks also have protection mechanisms against active attacks [26].

On the other hand present traffic classification methods can be grouped in three categories. Those are; flow-based, payload-based and host-based. Implementations of all three categories have their limitations. It is very hard when it comes to detecting new application traffic classification. The use of Traffic Dispersion Graphs (TDGs) will eliminate all above boundaries. Fig. 5 shows a TDG of a Peer-to-Peer Botnet.

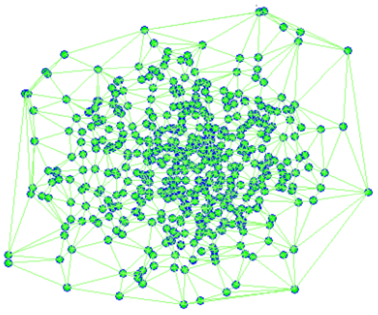


Fig. 5. Traffic dispersion graphs of a peer-to-peer Botnet

F. SMTP Traffic Analyzer

Due to their capability to automate large spam campaigns, Botnets are commonly used in the internet. Botnets transmit approximately 85% of the 100+ billion spam messages sent per day. There are several techniques to detect Spam messages. Those are content-based filtering, IP address of the sender and behavioral features like how the mail is sent.

SMTP Traffic Analyzer in Next Generation Security Framework is detecting spam messages and identifying Spam Hosts, Spamming IP Addresses, URLs associated with Spam messages and that information can be used with the Perimeter Filter to prevent from Botnets. SMTP Traffic Analyzer implemented using Postfix as the Mail Transfer Agent (MTA), Clam-AV as the virus scanner and SpamAssassin as the Spam filter.

VI. Conclusion

The next generation security framework is a security model to detect botnets on computer networks. This security model outlines how security is to be implemented on computer networks to detect botnets effectively. This research will be useful for, security researchers to have a look at a new model to detect botnet and everyone who is interest on security to have an in-depth knowledge on Botnets. The proposed Next Generation Security framework is based on passively monitoring network traffics. This model is based on the concept that multiple bots within the same Botnet will perform similar communication patterns and malicious activities.

The main highlighted point in our proposed detection framework from many other similar works is that, our proposed framework works as one general system for detection of Botnet. It focuses on IRC based Botnets, HTTP based Botnets, Peer-to-Peer based Botnets and Spam generated Botnets. In near future we will focus on reducing false positives generated from the system.

Acknowledgment

The authors would like to thank Prof. Gihan Dias, Dr. Chandana Gamage, Dr. Malaka Walpola and Dr. Shehan Perera for their invaluable supports in encouraging the authors to conduct MSc research and publish this paper.

References

[1] E. Gyu and Y. H. Shin, *A survey of Botnet: Consequences, Defenses and Challenges*.

[2] M. A. Rajab, J. Zarfoss, F. Monrose, and Andreas, "A Multifaceted Approach to Understanding the Botnet Phenomenon," in *Proc. the 6th ACM SIGCOMM Conference on Internet Measurement*.

[3] T. Holz, M. Steiner, F. Dahl, E. Biersacky, and F. Freiling, "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on StormWorm," in *The 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats.*

[4] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the Botnet phenomenon," in *Proc. 6th ACM SIGCOMM on Internet Measurement Conference, IMC 2006*, pp. 41-52, 2006.

[5] K. K. R. Choo, "Zombies and Botnets," *Trends and Issues in Crime and Criminal Justice*, Australian Institute of Criminology, Canberra, no. 333, March 2007.

[6] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage, "Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm," in *Proc. ACM* SIGOPS *OperatingSystem Review*, vol. 39, no. 5, pp. 148-162, 2005.

[7] F. Freiling, T. Holz, and G. Wicherski, "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks," in *Proc. 10th European Symposium on Research in Computer Security (ESORICS), Lecture Notes in Computer Science*, vol. 3676, pp. 319-335, September 2005.

[8] D. Dagon, C. Zou, and W. Lee, "Modeling Botnet propagation using time zones," in *Proc. 13th Network and Distributed System Security Symposium (NDSS'06)*, 2006.

[9] T. Holz, M. Steiner, F. Dahl, E. Biersacky, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: A case study on Storm-worm," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploitsand Emergent Threats*, 2008.

[10] Z. H. Chi and Z. X. Zhao, "Detecting and Blocking Malicious Traffic Caused by IRC Protocol Based Botnets," in *Proc. IFIP International Conference on* Network *and Parallel Computing Workshops*. pp. 485-489, 2007.

[11] 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,* 2008.

[12] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," in *Global Telecommunications Conference (GLOBECOM 2009),* Atlanta, USA.

[13] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting Botnets with Tight Command and Control," in *Proceedings of the 31st IEEE Conference on Local Computer Networks*, lcn, pp. 195-202, 2006.

[14] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer. "Using machine learning techniques to identify botnet traffic," in *Proceedings of the 2nd IEEE LCN Workshop on Network Security* (WoNS2006), 2006.

[15] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending against eclipse attacks on overlay networks," in *Proceedings of the 11th workshopon ACM SIGOPS European workshop*, 2004.

[16] B. N. Levine, C. Shields, and N. B. Margolin, *A Survey of Solutions to the Sybil Attack*.

[17] RFC 6335 - IETF Tools - Internet Engineering Task Force. [Online]. Available: http://tools.ietf.org/html/rfc6335

[18] RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1. [Online]. Available: http://tools.ietf.org/html/rfc2616

[19] Client–server model - Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Client-server_model

[20] C. Kalt, *Request for Comments (RFC) 2810: Internet Relay Chat – Architecture*, 2000.

[21] Overlay network - Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Overlay_network

[22] P. Porras, H. Saidi, and V. Yegneswaran. "A foray into Conficker's logic and rendezvous points," in 2nd *Proceedings of Usenix Workshop on Large Scale Exploits and Emergent Threats (LEET '09),* 2009.

[23] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. "Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience," in *Proceedings of ACM SIGCOMM*, Aug. 2003.

[24] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. "Secure routing for structured peer-to-peer overlay networks," in the *Proceedings of* SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pp. 299-314, 2002.

[25] S. Nagaraja, P. Mittal, C. Y. Hong, M. Caesar, and N. Borisov, "Bot Grep: Finding P2P Bots with Structured Graph Analysis," in *the Proceedings of USENIX Security Symposium*, August 2010.

Asanka Balasooriya received his BSc.Eng[Hons] degree in Computer Science and Engineering from University of Moratuwa in 2009. Present he is an MSc student in Information Systems Security in UoM. He is currently working as an Information Security Engineer in TechCERT.

Shantha Fernando is a senior lecturer at the Department of Computer Science and Engineering, UoM, and a Chartered Engineer. He received his BSc.Eng[Hons] degree in Computer Science and Engineering from University of Moratuwa in 1993. He started his career as a Software Engineer. He was attached to the private sector for 7 years, during which time he obtained his Master of Philosophy and then joined the academic service as a Senior Lecturer in the UoM in 2000. He obtained his PhD degree from Delft University of Technology, Netherland. Currently he serves as the Director, Engineering Research Unit of the UoM and as a Senior Lecturer at the Department of Computer Science and Engineering.