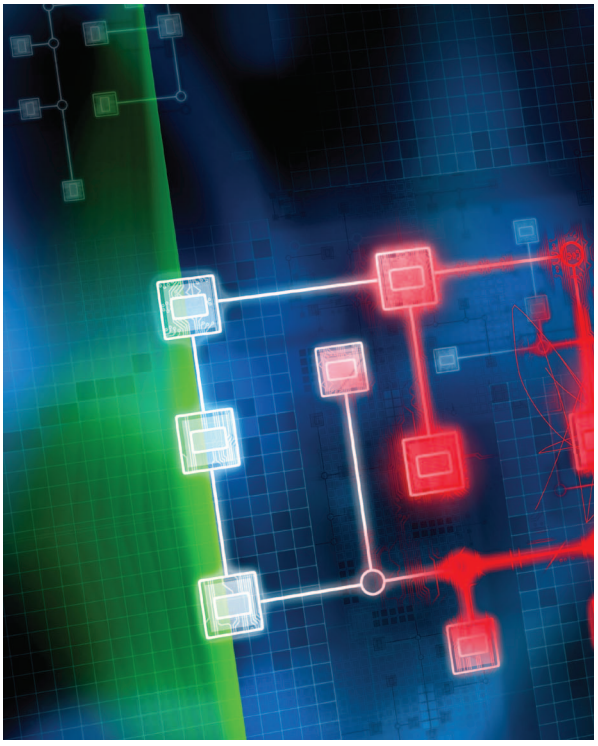

A Next-Generation Approach to Combating Botnets

Adeeb Alhomoud and Irfan Awan, *University of Bradford, UK*

Jules Ferdinand Pagna Disso, *EADS Innovations Works, Newport, UK*

Muhammad Younas, *Oxford Brookes University, UK*

As part of a defense-in-depth security solution for domain-controlled enterprise networks, a proposed self-healing system architecture is designed to increase resiliency against botnets with minimal disruption to network services.

Cybercrime has become the most lucrative global criminal activity, costing businesses, governments, and consumers an estimated $114 billion annually.[1] Consequently, protecting against cyberattacks is now a priority of many countries.

Botnets constitute a major tool for cybercriminals, giving them control over millions of computers. In 2012, Microsoft's Digital Crime Unit alongside the Financial Services—Information Sharing and Analysis Centre (FS-ISAC), the NACHA electronic payments association, and security consultancy Kyrus Tech made headlines by disrupting a huge botnet that was using Zeus malware.[2] However, such cross-industry actions are too expensive and complex to implement against all cybercriminals, who control as many as a quarter of the world's computers.[3]

A nature-inspired, self-healing architecture could enhance enterprise networks' resiliency to this growing threat.

UNDERSTANDING BOTNETS

The term botnet refers to a cluster of computers infected by the same malware. Each computer in this network serves as a software robot, or bot, capable of performing certain acts or executing commands automatically and repeatedly. In addition, a bot can simulate some human activities such as entering application login IDs.

In 1993, Jeff Fisher created the first bot as a useful feature in Internet Relay Chat to automate the maintenance and administration of IRC channels. Soon afterward, IRC bots emerged that were designed to quietly bypass computer firewalls and hijack system resources to carry out malicious tasks. Peer-to-peer (P2P) and HTTP-based bots, which are among today's most popular and effective bots, began to appear between 2000 and 2005.[4]

Bots can form a network of compromised computers or "zombies" subject to command and control by a bot master or "herder." A machine becomes compromised when a user downloads or opens malicious software.

Consider, for example, a botnet like Waledac, which was active between 2008 and 2010 and recruited some 80,000 bots through spam emails. Once a computer becomes infected, the botnet checks whether it has a public IP address. If the machine is behind a firewall, then the botnet makes it a *spammer*. If the computer can be accessed from the Internet, the botnet turns it into a *repeater* that relays requests between the botnet's spammers and back-end servers.

Published by the IEEE Computer Society

The botnet's repeaters and spammers continuously exchange lists of currently active repeaters. If a spammer loses communication with a repeater that goes offline, it tries to communicate with another repeater and update the lists. In addition, repeaters exchange lists of currently active back-end servers. The lists are signed with the bot master's private key so no other parties can insert their own back-end servers and take over the botnet.

Other botnets propagate in different ways—for example, via IM tools, operating system vulnerabilities, script injection, and P2P file shares.

Figure 1 shows a typical botnet's life cycle.[5] Once it infects a computer, a bot usually steals something such as personal information, authentication credentials, or credit card data. The machine then becomes part of the botnet, ready to perform designated malicious tasks. One function implemented in most botnets is the ability to participate in distributed denial-of-service (DDoS) attacks, in which bots flood a target system with User Datagram Protocol packets or Internet Control Message Protocol (ICMP) and SYN requests to deplete network bandwidth and system resources.[5] Other common functions include committing click fraud, hosting phishing and mule websites with rotating IP addresses, and compromising other computers with malware.[6-11]

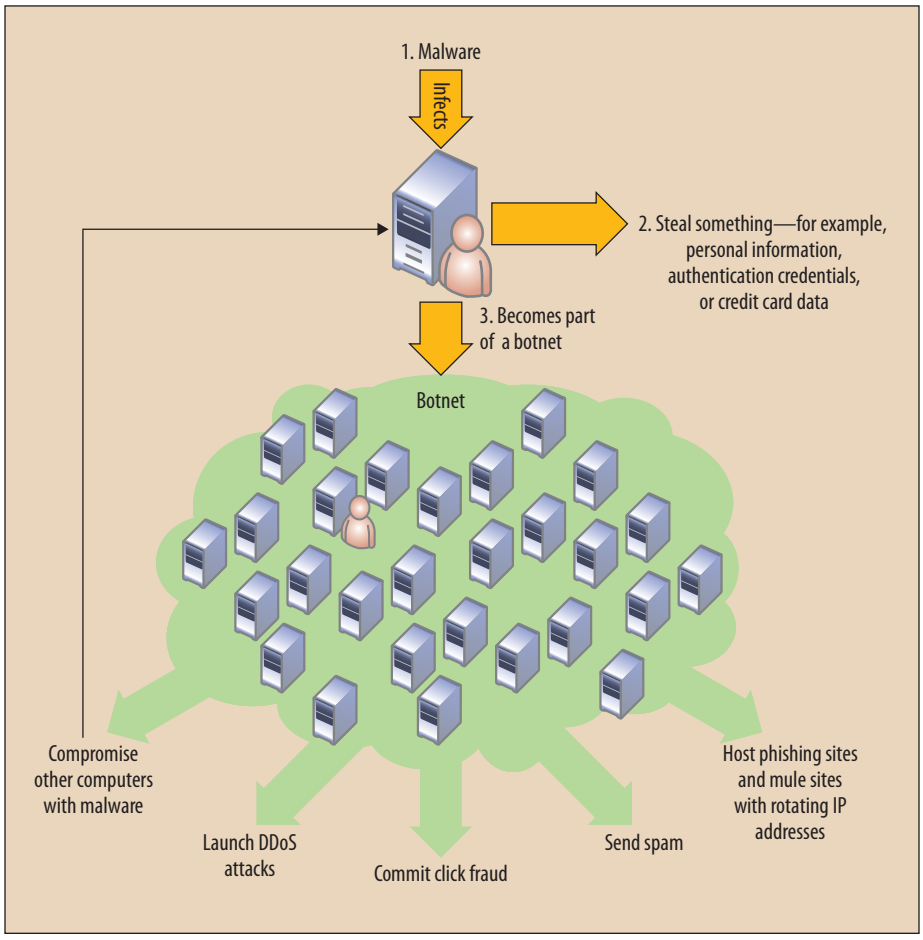

Figure 1. Botnet life cycle. Once it infects a computer, a bot usually steals something such as personal information, authentication credentials, or credit card data. The machine then becomes part of a botnet, ready to perform designated malicious tasks.

Botnets can mimic a genuine Simple Mail Transfer Protocol server or function as a ghost proxy server.[12] They also can silently upload to any system and quickly spread to infect thousands of machines. Coupled with the difficulty detecting them or locating their controllers, botnets' versatility and destructive capacity enable them to effectively disrupt cyberspace on a large scale, constituting serious challenges to researchers.

SELF-HEALING SYSTEMS

To help provide protection against botnets, we propose a self-healing architecture that enables networked systems to continually look for any alteration of "normal behavior" and apply appropriate corrective actions.

The self-healing computing system concept is inspired by the way organisms adapt to their environments by developing immunity against harmful viruses, bacteria, and toxins.[13] Researchers have investigated the nature-inspired self-healing paradigm in various areas. In operating systems, for example, developers have implanted interesting self-healing approaches such as code reloading, component isolation, and automatic restarts. In embedded systems, an example of a self-healing technique is running multiple instances of a task simultaneously with some idle tasks; in the event of failure, the system transfers the task's function to an idle task and executes it.[14]

A self-healing system can recognize when it is not operating correctly and, with little or no human intervention, either restore its normal working state or maintain system health until intervention occurs. In contrast to fault-tolerant systems, which employ stabilization and replication techniques, self-healing strategies generally involve isolating a

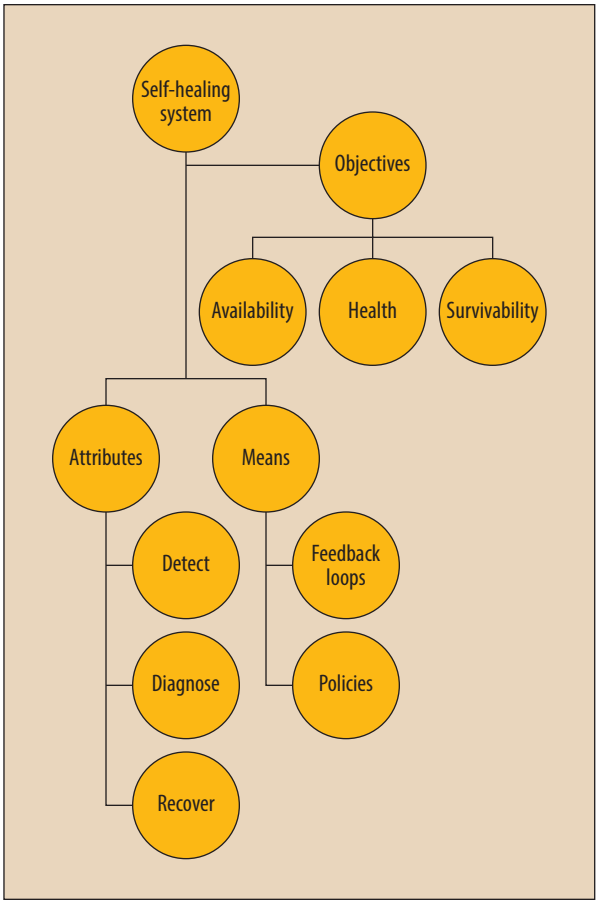

Figure 2. A self-healing system's main objectives are to maximize continuous availability, health, and survivability.

faulty component, taking it offline, and either fixing or replacing the failed component.[15]

As Figure 2 shows, a self-healing system's main objectives are to maintain continuous availability, health, and survivability.[16] To accomplish these goals, it has mechanisms to detect abnormalities in system functionality, including identifying the presence of foreign and potentially malicious agents, diagnosing the problem, and recovering the system—for example, by using redundant modular components to replace dead ones. Feedback loops and policies guide these mechanisms.

SYSTEM ARCHITECTURE

In line with the defense-in-depth concept, our self-healing system architecture is designed to mitigate the effects of any botnet infection of a workstation that might bypass traditional intrusion detection systems. It is based on a study of two HTTP-based botnets, Zeus and BlackEnergy, and two P2P botnets, Waledac and Storm.

The architecture is optimized for a domain-controlled network that connects users in departments and workgroups spread over an area ranging in size from the single floor of a building to a large geographic region. This type of network is common in hospitals, government agencies, universities, manufacturing centers, and large retail stores.

System components

The proposed architecture consists of five main modules.

Communication module. This module processes all communications to and from the self-healing system and also propagates the system.

Reporting module. This module creates log files of all system activity and sends alerts to the administrator for further investigation. It also generates daily, weekly, and monthly reports. A dedicated database archives all reports and alerts.

Detection module. This module checks all DNS host files and the network's inbound ports for any signs of botnet presence. The administrator can define the ports to be scanned; the module also can add its own ports by learning from the network—it will determine whether any application is using unique communication ports, such as accounting software, and add these to the white (trusted ports) list after confirming with the administrator. The detection module also takes MD5 checksums of the tcp.sys file, which is responsible for limiting socket connections and preventing the creation of raw TCP sockets and which some botnets can alter.

Healing module. This module attempts to remove a bot from any infected workstation. Each host on the network has an agent associated with it, and this module instructs these agents to perform certain tasks such as terminating a service to close a compromised port and stop the bot's communication ability. As a precautionary measure, if the system receives an alert indicating that an increasing number of computers in the network are using a new port, indicating a potential bot infection, the administrator can instruct the agents to block the port.

The healing module can also perform as a DNS sinkhole by spoofing the authoritative DNS server for malicious hosts. To register themselves or, in the case of some HTTP botnets, check for instructions, infected machines send a DNS query pointing to the botnet command-and-control IP address. The system can intercept this DNS query, obviating the need to add individual host entries when a malicious site changes its domain's hostname.

Control module. If the healing module fails, the control module will remove the infected node to avoid further botnet propagation through the network. It will send a simple ICMP echo request to the host and, if it receives no response, will consider it as "down" and send a report to the administrator requesting human intervention.

System operation

Figure 3 shows how the five modules interact. After

system initialization and propagation on a network workstation, the detection module scans the system. If the scan does not trigger an event, it will sleep for a designated time interval and then repeat the scan. If the scan does trigger an event, the detection module will report the incident to the administrator, who can activate the healing module. If the healing module is unsuccessful, the control module will attempt to take the infected machine offline and, if unable to do so, request human intervention.

Figure 4 shows a use case for the self-healing system. In this example, a botnet determines that an infected machine is behind a firewall, opens port 80, and modifies the computer's registry to make it a spammer. During a system scan, the detection module detects the open port and checks the registry, finding added values such as FWDONE, LastCommand, MyID, and Rlist. It reports the results to the administrator, who activates the healing module. The healing module closes the open port and modifies the registry. Feedback indicates that these processes were successful, and the healing module generates a report for the administrator.

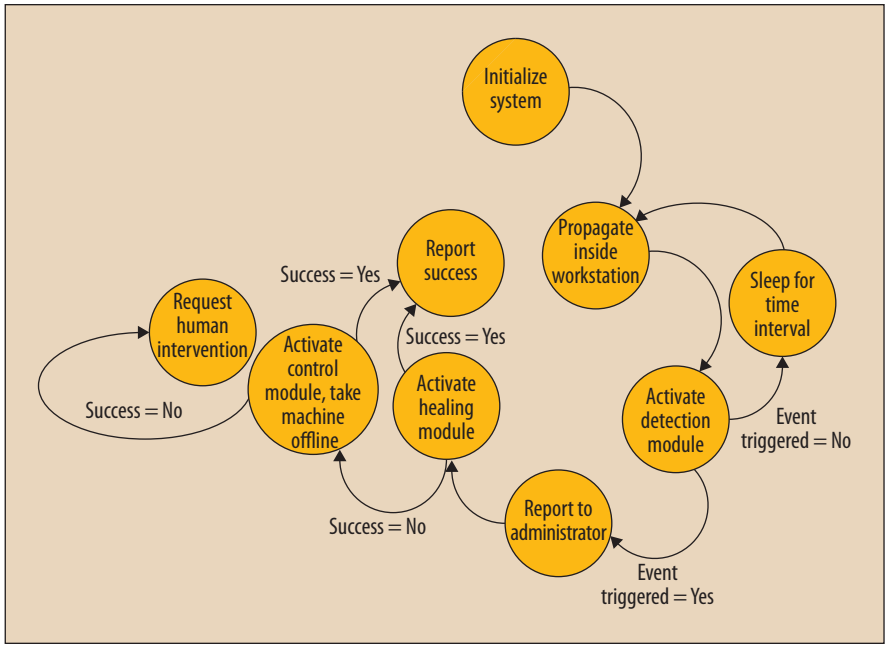

Figure 3. Self-healing system architecture.

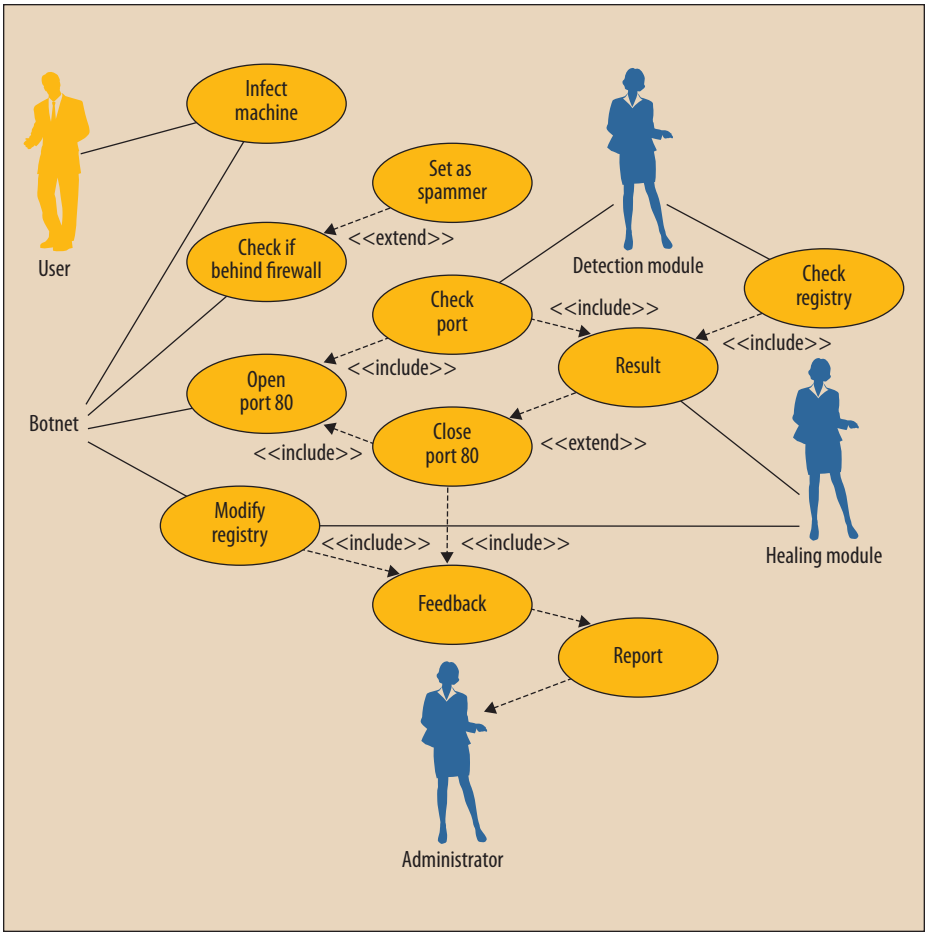

Figure 4. Self-healing system use case.

Cyberattacks are continuously evolving and becoming increasingly problematic for IT personnel. Cybercriminals use botnets to destroy or degrade critical networks or systems and to gather information not only from personal computers but also from companies, government organizations, and large institutions such as hospitals and banks.

As part of a defense-in-depth security solution for domain-controlled enterprise networks, our proposed self-healing system architecture is designed to increase resiliency against botnets with minimal disruption to network services. We continue to research various types of botnets and derive appropriate use cases to improve the functionality of the various system components. Our modular approach makes it possible to modify existing components and add new ones without changing the original architecture.

References

1. Communication from the European Commission to the Council of the European Union and the European Parliament, "Tackling Crime in Our Digital Age: Establishing a European Cybercrime Centre," 28 Mar. 2012; http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2012:0140:FIN:EN:PDF.
2. Microsoft, "Microsoft Joins Financial Services Industry to Disrupt Massive Zeus Cybercrime Operation that Fuels Worldwide Fraud and Identity Theft," press release, 25 Mar. 2012; www.microsoft.com/en-us/news/press/2012/mar12/03-25CybercrimePR.aspx.
3. T. Weber, "Criminals 'May Overwhelm the Web,'" *BBC News*, 25 Jan. 2007; http://news.bbc.co.uk/2/hi/business/6298641.stm.
4. J.-S. Lee et al., "The Activity Analysis of Malicious HTTP-Based Botnets Using Degree of Periodic Repeatability," *Proc. Int'l Conf. Security Technology* (Sectech 08), IEEE CS, 2008, pp. 83-86.
5. Organisation for Economic Co-ordination and Development, *Malicious Software (Malware): A Security Threat to the Internet Economy*, ministerial background report, 2008; www.oecd.org/internet/ieconomy/40724457.pdf.
6. N. Ianelli and A. Hackworth, "Botnets as a Vehicle for Online Crime," CERT Coordination Center, 1 Dec. 2005; www.cert.org/archive/pdf/Botnets.pdf.
7. M.T. Banday, J.A. Qadri, and N.A. Shah, "Study of Botnets and Their Threats to Internet Security," *Sprouts: Working Papers on Information Systems*, vol. 9, no. 24, 2009; http://sprouts.aisnet.org/594/1/Botnet_Sprotus.pdf.
8. R. Puri, "Bots & Botnet: An Overview," white paper, SANS Inst., 8 Aug. 2003; www.sans.org/reading_room/whitepapers/malicious/bots-botnet-overview_1299.
9. J. Nazario, "Botnet Tracking: Tools, Techniques, and Lessons Learned," presentation, 2007 Black Hat USA Conf.; www.orkspace.net/secdocs/Conferences/BlackHat/Federal/2007/Botnet%20Tracking%20-%20Tools,%20Techniques,%20and%20Lessons%20Learned-paper.pdf.
10. A. Dainotti et al., "Analysis of a '/0' Stealth Scan from a Botnet," *Proc. Internet Measurement Conf.* (IMC 12), ACM, 2012; www.caida.org/publications/papers/2012/analysis_slash_zero/analysis_slash_zero.pdf.
11. W. Lu, M. Tavallaee, and A.A. Ghorbani, "Automatic Discovery of Botnet Communities on Large-Scale Communication Networks," *Proc. 4th ACM Symp. Information, Computer, and Comm. Security* (ASIACCS 09), ACM, 2009; doi:10.1145/1533057.1533062.
12. G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," *Proc. 15th Ann. Network and Distributed System Security Symp.* (NDSS 08), 2008; http://corescholar.libraries.wright.edu/cse/7.
13. J. Twycross and U. Aickelin, "An Immune Inspired Approach to Anomaly Detection," arXiv preprint arXiv:0910.3117, 2009; http://arxiv.org/abs/0910.3117v1.
14. M. Glass et al., "Reliability-Aware System Synthesis," *Proc. Design, Automation & Test in Europe Conf. & Exhibition* (DATE 07), European Design and Automation Assoc., 2007; www.date-conference.com/proceedings/PAPERS/2007/DATE07/PDFFILES/03.4_6.PDF.
15. A.G. Ganek and T.A. Corbi, "The Dawning of the Autonomic Computing Era," *IBM Systems J.*, vol. 42, no. 1, 2003, pp. 5-18.
16. M. Salehie and L. Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges," *ACM Trans. Autonomous and Adaptive Systems*, vol. 4, no. 2, 2009, article 14.

Adeeb Alhomoud *is a PhD student in the School of Computing, Informatics and Media at the University of Bradford, UK. His research interests include botnets and malware propagation. Alhomoud is a member of IEEE. Contact him at a.m.alhomoud@student.bradford.ac.uk.*

Irfan Awan *is a professor of computer science in the School of Computing, Informatics and Media at the University of Bradford, UK. His research interests include network security, communication systems, and performance modeling. Awan received a PhD in computer science from the University of Bradford. He is a member of IEEE and the British Computer Society, and a fellow of the Higher Education Academy. Contact him at i.u.awan@bradford.ac.uk.*

Jules Ferdinand Pagna Disso *heads the Cyber Security Research Lab at EADS Innovation Works, Newport, UK. His research interests include cybersecurity for industrial control systems, botnets, cloud security, forensics, threat analysis, and vulnerability identification. Pagna Disso received a PhD in intrusion detection systems from the University of Bradford. He is a member of IEEE and ACM. Contact him at julesferdinand.pagna@edas.com.*

Muhammad Younas *is a senior lecturer in computing at the Department of Computing and Communication Technologies, Oxford Brookes University, UK. His research interests include Web and Internet technologies, service-oriented computing, and pervasive and mobile information systems. Younas received a PhD in computer science from the University of Sheffield, UK. He is a member of the IEEE Computer Society. Contact him at m.younas@brookes.ac.uk.*