Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

A Large-Scale Zero-Day Worm Simulator for Cyber-Epidemiological Analysis

Luc Tidy, Steve Woodhead and Jodie Wetherall

***Abstract*—The cost of a single zero-day network worm outbreak has been estimated at US$2.6 billion. Additionally zero-day worm outbreaks have been observed to spread at a significant pace across the global Internet, with an observed infection proportion of more than 90 percent of vulnerable hosts within 10 minutes. The threat posed by such fast-spreading malware is therefore significant, particularly given the fact that network operator / administrator intervention is not likely to take effect within the typical epidemiological timescale of such infections.**

An accepted tool that is used in researching the threat presented by zero-day worms is the use of simulation systems. However when considering zero-day worm outbreaks on the Internet there are persistent issues of scale and fidelity. The Internet Worm Simulator (IWS) reported in this paper is designed to address these issues by presenting a novel simulation method that, on a single workstation, can simulate an entire IPv4 address space on a node-by-node basis. Being able to simulate such a large-scale network enables the further analysis of characteristics identified from worm analysis. As IWS does not rely on mathematical approximation, the epidemiological attributes identified from real-world data can be tested for zero-day worm outbreaks on the Internet.

Experimentation indicates that IWS is able to accurately simulate and corroborate with reported characteristics of two previous zero-day worm outbreaks. It is intended that, in future, IWS may be used to aid both in the analysis of previous worm outbreaks and the testing of hypothetical zero-day worm outbreak scenarios.

***Keywords*—cyber defence, malware, network worm, simulation, zero-day worm.**

I. INTRODUCTION

A zero-day worm is a type of malware that exploits a vulnerability that has not been patched or acknowledged at the point of an outbreak, which owing to an automatic propagation method can spread pervasively throughout a network; which is exacerbated by either a lack of detection or speed of propagation [1]. In order to tackle such outbreaks an understanding of how they occur, their propagation method, and their epidemiological characteristics across a given network is essential [2]. Worms are often hard to prevent, counter, or contain, primarily owing to their potential speed of propagation; raging from fast random-scanning worms to slower 'stealthy' worms that employ various techniques to propagate undetected. In order to aid the analysis of zero-day worm epidemiology analytical models, such as [3], and simulation systems such as [4], have been adopted.

Since the widespread worms that occurred in the first four years of the 21st century, such as Code Red [3] and Slammer [5], there have been a series of zero-day worms such as Conficker [6] in 2007 and Duqu [7] in 2011. These can incur significant costs, with one estimate for the cost of the Code Red outbreak being US$2.6 billion. It is imperative that defences are employed in order to mitigate or prevent such worm outbreaks. Simulation systems provide a tool which can aid research into worm analysis [5], and network worm countermeasures [8,9].

Issues persist in being able to simulate a worm outbreak on the Internet, owing to issues of scale and fidelity. Previous work has focused on methods to reach this large-scale by either using mathematical approximations about how a worm has spread [3], or by introducing a mixture of detailed packet-level simulation and mathematical approximations [10].

The remainder of this paper is presented as follows: Section 2 sets out a definition of terms; Section 3 discusses the relevant previous work; Section 4 details the design of the Internet Worm Simulator (IWS); Section 5 presents the experimental methodology and results; and finally section 6 concludes the paper with a discussion summarising the findings and identifying any limitations and future work.

II. LEXICON

A lexicon has been presented for the clarification of the following terms, owing to their specific use in this paper.

Zero-Day Worm: In this paper this is defined as a type of malicious software that propagates automatically without human interaction, using a vulnerability that has not been patched or widely acknowledged at the point of an outbreak. This is a similar definition to the taxonomy described by Weaver et al. [2], and other published literature (see [3, 5, 11]).

Epidemiological Analysis: In this paper an epidemiological analysis is based on the rate at which susceptible nodes become infected. This paper focuses on the total number of infected nodes at any given point during an outbreak, and not the difference in the number of newly infected nodes at each period of time.

Large-Scale Simulation: This paper defines large-scale simulation as being able to, or the attempt to, simulate an entire IPv4 address space of nodes for a given scenario; often with the intent of simulating a network the size of a contemporary Internet.

Luc Tidy, Steve Woddhead and Jodie Wetherall
Internet Security Research Laboratory.
University of Greenwich,
United Kingdom

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

Datagram: In accordance with RFC1594 [12] this paper defines a datagram as a data entity that is to be transmitted across a network, specifically, this term is used to describe the number of data entities transmitted by a worm during an outbreak.

State Variable Machine: A processing method in which data object interaction is derived from a set of discrete states. In this paper this is focused on using a set of discrete states in order to describe the interaction between network nodes for the purpose of zero-day worm analysis.

III. RELEVANT PREVIOUS WORK

Owing to brevity, this paper focuses on only the key areas of previous literature. The key areas of focus in this paper can be divided into two key domains: the use of simulation systems in worm analysis; and the methods employed in order to simulate large-scale networks.

A. Simulation Systems in Worm Analysis

A number of simulation systems have been used in order to undertake zero-day worm analysis, most notable of which is [3],that used simulation in order to analyse the epidemiology of the Code Red outbreak of 2001. This has been extended in other works, such as that by Moore et al. [5], who used a random-constant spread model to analyse the Slammer outbreak of 2003. These simulations rely on generalised mathematical models in order to simulate the scale of the IPv4 Internet..

B. Large-Scale Simulation

In order to improve on generalise mathematical modelling techniques, packet-level simulators, such as GTNetS [4], have been developed to increase the granularity of the simulation but at the cost of a substantial overhead. This means that although the issue of fidelity is tackled, this is at the cost of not being able to feasibly reach sufficient scale for simulating zero-day worm outbreaks on the IPv4 Internet. Two approaches have been proposed to address these competing limitations: hybrid simulation methods; and state variable machine methods.

Hybrid simulation methods, [10] combine the granularity of packet-level simulators alongside the generalised mathematical methods in order to reach a larger network scale. Reported research in this area is limited however, with significant issues to be resolved.

State variable machine methods, such as those employed by Ediger [1] and Wei et al. [13] tackle issues of scale by reducing each node to a state variable machine. This means certain characteristics about each node can be tracked without the significant overhead from packet-level simulation. In both [1, 13] the approach taken means that a single node is represented by a software object in the order of kilobytes in size. This significant memory allocation means that neither can reach the scale of an entire IPv4 address space on a node-by-node basis.

By reducing the memory allocation required for each node, IWS is able to retain node-by-node granularity while still being able to simulate networks with a size as large as the IPv4 address space.

C. Motivation

As far as the authors are aware no previous worm simulation tool has been reported which is able to simulate an entire IPv4 network on a node-by-node basis. Of note is the simulator reported in [1] which attempts this however, it only reaches autonomous-system level granularity. It is intended that by being able to simulate on a node-by-node level of granularity the simulator will enable the accurate epidemiological analysis of large-scale zero-day worms. This will mean simulation is undertaken based on the observed, or hypothesised, characteristics of a worm without having to employ a generalised mathematical approximation.

The simulator reported in this paper meets these requirement, as well as the target of running on a single PC workstation.

IV. DESIGN AND CAPABILITIES

The Internet Worm Simulator (IWS) was developed with the intention of employing it in an investigation of the epidemiology of existing and hypothetical worms. The ability to retain node-by-node granularity and scalability were key design criteria. Similarly the ability to simulate a broad range of worm spreading algorithms was a key requirement. Based on the findings from a review of existing simulator designs, along with preliminary work with the NWS simulator reported in [3], it was decided the simulator would be developed using the SVM method. This is owing to the fact that it would enable the simulation of the whole IPv4 Internet address space, whilst maintaining node-by-node granularity.

IWS has been developed in the C programming language, using the GCC compiler under a 64 bit 3.4 Linux Kernel. IWS only requires one byte of memory per host; this means for an IPv4 sized network it has a memory footprint of 4GB. This enables iws to meet the design requirements, owing to the use of a compiled language that offers good low level control of memory resources. These specifications mean that IWS can be ported to other 64 bit operating systems, such as Windows, and can also be run on a variety of different hardware configurations.

In order to minimise the memory requirements of the simulator, node states are stored in a single byte of memory. Compared to NWS this is a significant reduction, where in experimentation carried out by the others NWS typically used 1.6KB per node. The SVM states will vary dependent on the worm instance being simulated however, this allows a variety of variables such as tracking there infection status and available bandwidth. These states can be modified depending on whether or not it is appropriate for each simulation respectively.

The IWS simulation method can be divided into three key areas: initialisation engine, infection engine, and reporting engine. As Fig. 1 illustrates these engines pass data between

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

them in order to produce the results. The engines can be modified to incorporate different parameters, node compositions, algorithms, or reporting format according to the required specification.

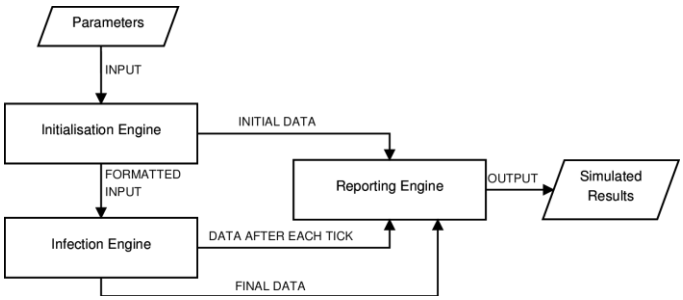

Figure 1 IWS Design

Having received the parameters that are to be used in a given simulation, the initialisation engine begins by setting up the states of each node. This step will define which nodes are susceptible to infection and which are initially infected. If other states are employed this would also be set by the initialisation engine, with the intent of preparing the parameters provided for simulation. The final step undertaken by the initialisation engine is to pass its details to the reporting engine having completed its other steps.

The infection engine, having received the formatted data from the initialisation engine uses this in order to carry out the worm infection algorithm provided. This is an iterative process that uses a "clock tick" in order to determine the total number of infected hosts at a given point in time. For this paper a clock tick is representative of a single simulated second, and as such is passes on the current state of the worm outbreak to the reporting engine in a format that is representative of each simulated second. Upon a set point, for example either a particular amount of simulated time passing or the number of infected nodes reaching a set threshold, the infection iterations cease and the final data is passed to the reporting engine.

In order to log all the activity in a desirable output, the reporting engine takes data from both the initialisation and infection engines . This means the number of nodes with a certain state can be logged, giving the opportunity for comparative and exploratory investigations and epidemiological analysis.

IWS can be modified for a variety of different scenarios using a series of user-defined parameters that detail; the worm propagation method, the network composition, the worm packet size, its initial infected hosts at the point of an outbreak, and the total number of susceptible hosts..

V. EXPERIMENTATION

The selection of experimental results reported in this paper focus on the operational evaluation of the simulator. In order to do so simulation was undertaken in order to compare its results with empirical data of previous worm outbreaks. For this paper the Slammer worm of 2003 and the Witty Worm of 2004 have been used for comparison, using the empirical data reported in [5,11].

As both these worms performed random-scanning as part of their infection algorithm, a range of simulation results have been presented. A set of five simulations, using different pseudo-random number seedshave been undertaken.

A. *Methodology*

In order to assess the capabilities of IWS it was validated against the empirical data presented in [5,11], for the Slammer and Witty worms respectively.

Firstly, the average reported datagram transmission rate has been simulated in order to represent a preliminary comparison with the empirical data. Secondly, further worm characteristics are included in order to test the accuracy of the simulator.

Finally, in order to assess the ability of iws to simulate a large-scale infection, a hypothetical scenario has been simulated wherein all versions of a popular operating system are vulnerable to a Slammer-like worm infection vector.

B. *Slammer*

The analysis undertaken by Moore et al. [5] reports a set of key characteristics about the Slammer worm outbreak of 2003, which have been used as the simulator parameters. Moore et al. report that approximately 18 hosts per million of the entire IPv4 address space were susceptible to infection. The maximum recorded datagram scanning rate was observed at 26,000 datagrams per host per second, which is reasonable given a common upper bound of 100BaseT interfaces, and a total worm Ethernet frame size of 430 bytes.

It is reported that Slammer exhibited a mean of 4,000 datagrams generated per infected host per second throughout its outbreak period, meaning that the mean transfer speed per node was just over 13 megabits per second. Plotting this mean value against the data reported by Moore et al., as shown in Fig. 2, it can be seen that the simulator produces, on average, fewer datagrams generated per second. This is potentially due to the influence of the peak scanning rate, having a greater number of datagrams at this early stage could significantly impact the rate at which new hosts become infected.

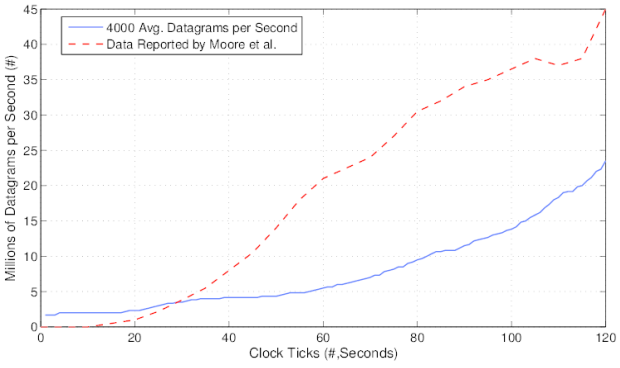

Figure 2 Preliminary IWS Slammer Worm Simulation

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

$$r_t = f r_{t-1} \quad (1)$$

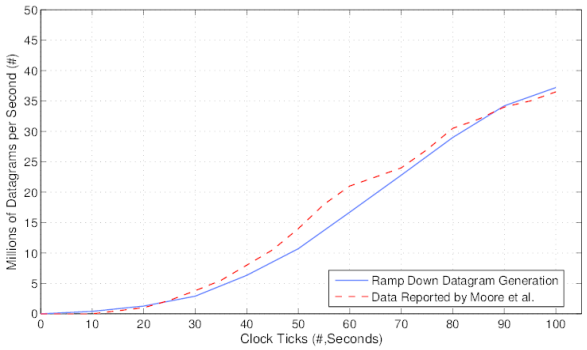

Figure 3 IWS Slammer Worm Simulation

Taking this into consideration, a ramp-down factor has been applied based on the reported findings by Moore et al. that an initial spike in the datagrams per second observed reduced to a lower average overall owing to an increase in congestion as more datagrams were generated. This higher datagram generation rate and datagram generation decrease has been applied so that each subsequent clock tick has a ramp-down factor (f) of 0.95, where the datagram generation rate, r, for a given clock tick, t, is determined according to (1). Fig. 3 illustrates that by applying this factor, the simulation matches the datagram generation rate more closely for the Slammer worm, whilst still operating on a node-by-node basis.

C. *Witty*

Key characteristics have also been reported by Shannon et al. [11] regarding the Witty worm outbreak of 2004 that has been incorporated in the simulator. It was reported to have a much smaller susceptible population than Slammer, recorded as 12,000 hosts, or between 2 and 3 hosts per million of the entire IPv4 address space. Unlike Slammer, Witty is also reported to have had a variable datagram size, with an Ethernet frame size of between 796 and 1307 bytes.

With a frame size just over three times larger than Slammer, it is reported that Witty was able to maintain an average of 370 datagrams generated per infected host per second, with a main peak of 970. Of note here is that 38 nodes were reported to be transferring at 9,700 datagrams per second continuously for over an hour. Simulating the average values as 3 hosts per million susceptible, Fig. 4 shows that the reported metrics do fall between the two datagram size values, with a sharper rise at the start of the outbreak, potentially owing to the 38 hosts generating datagrams at ten times the main peak level.

By including the set of 38 hosts as part of the infected hosts, I, for a given clock tick, t, where nodes h_1 to h_{38} generate at 9,700 datagrams per second and the other nodes h_{39} to $h_n$scan at 970 datagrams per second the faster increase in the number of infected hosts can be included, as shown by the matrix in (2). Fig. 5 on page 1 illustrates how by including this node differentiation a closer match to the empirically reported Witty worm behaviour is achieved.

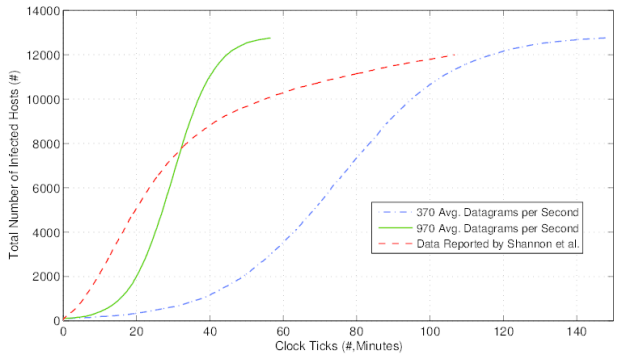

Figure 4 Preliminary IWS Witty Worm Simulation

$$I_t = \left[h_1 \ldots h_{38} \quad h_{39} \ldots h_n \right] \quad (2)$$

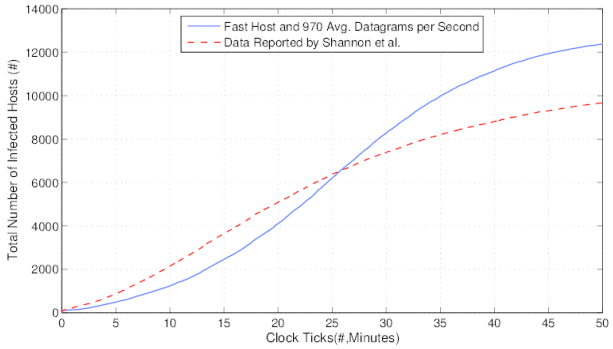

Figure 5 IWS Witty Worm Simulation

D. *Large-Scale Simulation*

In order to demonstrate the scalability of iws a worm outbreak scenario that is much larger than that previously reported has been simulated. This scenario uses the same average empirical metrics of the Slammer worm outbreak however, a simulation with a significantly larger susceptible populous has been undertaken.

The scenario of an exploit that is common to all recent Windows operating systems has been considered. W3Counter[14] reports that 78.8% of internet-connected hosts use a recent version of the Windows operating system (7, Vista or XP) and so a susceptible populous of 80% has been simulated.

Fig. 6 demonstrates that with such a large number of hosts susceptible a fast-scanning worm like Slammer may spread incredibly quickly; in this case only taking 1.5 seconds for 99% of the susceptible populous to become infected on the current IPv4 Internet.

VI. DISCUSSION

The cyber-epidemiological analysis of zero-dayworms on the Internet remains a significant challenge, and the use of simulation systems remain a viable tool for such research. This paper has presented a novel network worm simulator, the

ISBN: 978-981-07-5461-7

Internet Worm Simulator (IWS), and has demonstrated its feasibility for the simulation of zero-day worm epidemiology in large-scale networks (currently up to 2^{32} hosts). It has also confirmed the accuracy of the simulator, by comparing its output with available empirical data for real-world zero-day worm outbreaks.

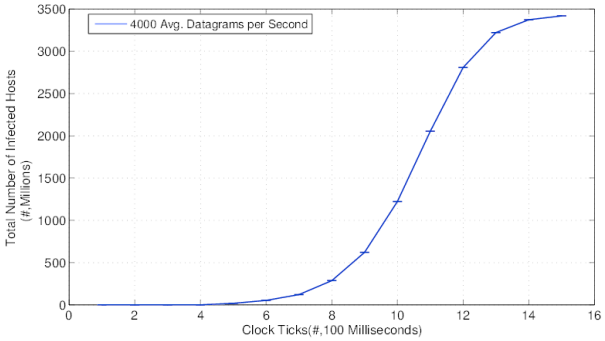

Figure 6 Hypothetical Slammer-style Worm with an 80% Susceptible Populous

In comparison with other worm simulation techniques, such as packet-level simulation, statistically-based generalised mathematical modelling and a hybrid of the two, the state-variable machine method employed in IWS can be used to simulate zero-day worm epidemiology on an Internet-scale network, whilst still retaining node-by-node granularity. This paper has demonstrated that the method can incorporate more complex epidemiological behaviour, as well as utilising the node-by-node granularity it offers in order to achieve more accurate simulation results.

It is hoped that the IWS will form a useful tool for the further wide ranging epidemiological investigations of both real and hypothetical zero-day worms and countermeasure techniques,, providing security researchers and professionals with the capability to simulate large-scale outbreaks on non-specialist, widely available computer systems whilst retaining significant accuracy.

A. *Limitations and Future Work*

In this paper the IWS relies on the existing worm analysis of previous worm outbreaks, which in turn relies on accurate data being gathered at the point of an outbreak. If there is an issue with the data gathering process, as shown in [5] as the Slammer worm outbreak matures, then the reliability of any worm analysis becomes an issue. Similarly, this does not account for hypothetical scenarios, or scenarios where only a limited set of information is available to provide the simulator with parameters, which may mitigate some issues regarding incomplete empirical data.

This paper has reported work undertaken as part of a development programme for the IWS. Currently, the authors expect to extend the work in a number of areas, with a view to further enhancing the usability and capabilities of the simulator. Key areas in which further work is currently focused are as follows:

- Decreasing the execution time of the simulator while retaining the single workstation design criteria.
- Improvements in simulator accuracy and granularity, in particular by including a more granular model of the network topology and bandwidth constraints, whilst retaining, in so far as possible, the simple set up of the simulator.

Acknowledgment

Additional Witty Worm Data provided by: The CAIDA UCSD Dataset on the Witty Worm - March 19-24, 2004, http://www.caida.org/data/passive/witty_worm_dataset.xml.

References

[1] Ediger B: *Simulating Network Worms - NWS Network Worm Simulator*. http://www.stratigery.com/nws/, 2003. Accessed 6th May 2011.
[2] Weaver N, Paxson V, Staniford S, Cunningham R: "A Taxonomy of Computer Worms", , pp. 11-18, 2003.
[3] Zou CC, Gong W, Towsley D: "Code Red Worm Propagation Modeling and Analysis", , pp. 138-147, 2002. Accessed 6th May 2011.
[4] Riley G: "The Georgia Tech Network Simulator", , pp. 5-12, 2003.
[5] Moore D, Paxson V, Savage S, Shannon C, Staniford S, Weaver N: "The Spread of the Sapphire/Slammer Worm", *IEEE Security and Privacy*, pp. 33-39, 2003. Accessed 6th May 2011.
[6] Shin S, Gu G: "Conficker and beyond: a large-scale empirical study", , pp. 151—160, 2010.
[7] Faisal M, Ibrahim M: "Stuxnet, Duqu and Beyond", *International Journal of Science and Engineering Investigations*, pp. 75-78, 2012.
[8]Whyte D, Kranakis E, van Oorschot PC: "DNS-Based Detection of Scanning Worms in an Enterprise Network", , 2005.
[9] Wong C, Bielski S, Studer A, Wang C: "Empirical Analysis of Rate Limiting Mechanisms", , pp. 22-42, 2006. Accessed 7th May 2011.
[10] Liljenstam M, Yuan Y, Premore BJ, Nicol D: "A Mixed Abstraction Level Simulation Model of Large-Scale Internet Worm Infestations", , 2002.
[11] Shannon C, Moore D: "The Spread of the Witty Worm", *IEEE Security and Privacy*, pp. 46-50, 2004
[12] Marine A, Reynolds J, Malkin G: *RFC1594*. 1994. URL https://tools.ietf.org/html/rfc1594.Accessed 15th November 2012.
[13] Wei S, Mirkovic J, Swany M: "Distributed Worm Simulation with a Realistic Internet Model", , pp. 71-79, 2007. Accessed 7th May 2011.
[14] Awio Web Services LLC: *W3Counter - Global Web Stats*. 2012. URL http://www.w3counter.com/globalstats.php. Accessed November 2012.

About Author (s):

Luc Tidy obtained his BEng Honours degree in Computer Systems and Software Engineering in 2010. He is currently a Research Assistant and Doctoral Candidate in the Internet Security Research Laboratory, University of Greenwich. His research is centred on the development of cyber-epidemiological analytical systems for large-scale, zero-day worm outbreaks

Steve Woodhead obtained his BSc Honours degree in 1987 and his PhD in 1992, both in Electronic Engineering. He currently holds the position of Reader in Computer Systems and Networks in the Department of Computer and Communications Engineering, University of Greenwich.. Dr Woodhead has published over 50 research papers in refereed journals and conferences

Jodie Wetherall obtained his BEng Honours degree in Internet Technologies in 2001, and his PhD in Computer Engineering in 2010. He currently holds the position of Programme Leader for Games and Entertainment Systems Software Engineering in the Department of Computer and Communications Engineering, University of Greenwich.