Gotta Catch 'Em All! Innoculous: Enabling Epidemiology of Computer Viruses in the Developing World

Michael Paik
New York University
New York, NY
United States
mpaik@cs.nyu.edu

ABSTRACT

Computer users in the developing world are faced with myriad challenges, from limited bandwidth to higher costs of usage and ownership. However, among the most pernicious problems is the prevalence of computer viruses, which have immediate and unexpected economic costs, often to those who are least able to bear the burden of such costs.

While statistics are available for virus infection rates, these rates only reflect reports from legally purchased copies of antivirus software run on internet-connected machine, and not the preponderance of software in the developing world, which is illegally obtained, out of its license period, or operated offline and therefore not updated. Anecdotal evidence on the ground indicates infection rates an order of magnitude higher, indicating a dearth of accurate information.

In this paper I present Innoculous, a system consisting of a specially crafted USB key, software, and an incentivization strategy aimed towards disinfecting infected machines, creating revenue streams for small businesses and individuals in the developing world, and obtaining rich information about computer virus infections in the environment in question.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Invasive Software*; K.4.f [**Computing Milieux**]: Computers and Society—*Security*

General Terms

Security, Measurement, Design

Keywords

Viruses, Epidemiology, Data Collection

NSDR'11, June 28, 2011, Bethesda, Maryland, USA

1. INTRODUCTION

The developing world faces many challenges in the use of technology, some of them endemic to the socioeconomic and educational state of people at the base of the pyramid, and others that are merely artifacts of the high relative cost of technology and dearth of technology education in that environment.

Among those in the latter class is the problem of computer virus infections in existing installations of both online (cybercafes, universities) and offline (copy shop kiosks, primary and secondary schools, private homes) computers. Various prior work has posited that virus infections are a clear and present problem [17–22, 25] in the developing world, but it is difficult to pin down reliable figures about the rates and types of infections, as well as the scale of damage done.

While figures do exist [23] for infection rates from, e.g. Kaspersky Labs [1] and McAfee [2], these rates are in comparison to global infection reports, and not in comparison to the installed base of computers. Moreover, these statistics do not include infections that may have been detected by other antivirus products, particularly those that have been pirated or used offline, or indeed those which have gone undetected due to old virus definitions or lack of virus protection altogether.

The figures that do exist however, reflect massive growth in the absolute number of infections, with Kaspersky indicating an order of magnitude increase in e.g. East Africa between July 2009 and September 2010, with a fourfold increase in the months from March to September alone [23].

Anecdotal accounts by experts on the ground put the figure of infection rates in the developing world at up to 80% [24], indicating a well and truly endemic problem, a figure corroborated by recent surveys by Bhattacharya et al. [18] conducted in Bangalore, India. These survey data also indicate an extremely high rate of software piracy, which is further corroborated by annual studies by the Business Software Alliance and the International Data Corporation [11], which estimate piracy rates in the developing world at between 65% (India) and 92% (Zimbabwe) as of 2009. This rate of software piracy and the lack of updates and security patches it implies exacerbate the infection rate, as well as obfuscating true virus infection rates as antivirus solutions are patched not to dial home with registration information as well as infection statistics. In addition, malware authors are known to distribute their software in infected versions of popular pirated software [8].

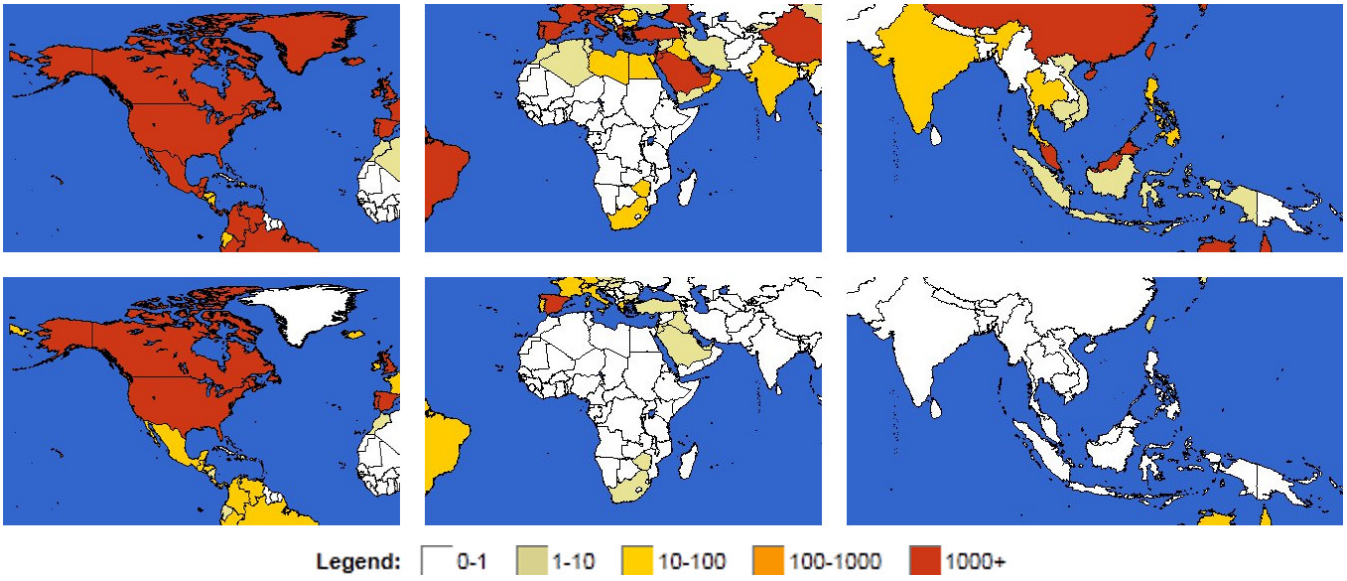

Figure 1: Virus infection maps from [3]. From left to right, North America, Africa, and Southeast Asia. Top row shows overall infection rates per million citizens, bottom row shows infection rates from the top 10 infections.

In this paper I present Innoculous, a software system combined with a specially crafted USB memory key that, together with an incentivization strategy, is designed to both help alleviate virus infections, particularly on computers that are not internet-connected, as well as gather detailed information about the epidemiology of viruses and their overall ecosystem in the developing world: strains, infection rates and vectors, correlated infections, etc.

1.1 Motivation - Virus Ecology

While the data available, at least to the public, are not extremely detailed, certain aggregate data are, including the maps shown in Figure 1. The top row of Figure 1 displays virus infections per million citizens according to McAfee [3], while the second row shows infections from the top 10 viruses. While data aggregated at this level is inconclusive, the difference between North America and the developing regions in this regard is remarkable in that it strongly suggests that the specific virus types present in the developing world, while high in absolute infection rate (e.g. in Africa, one third of all files reported scanned were infected with something), display a different ecology than in the developed world.

While it is far from clear that these data represent anything approaching a representative cross-section, as those willing and able to pay for McAfee subscriptions are likely from the middle and upper classes, the data suggest a gap in our understanding of the viral ecology and epidemiology outside the developed world.

2. DESIGN

The inception of Innoculous as a system to both clean virus infections and record data about these infections was, in part, inspired by Disk Knight [4]. Disk Knight is a USB worm which was designed to spread along the same infection vectors as traditional USB viruses and worms, but with a 'beneficial' payload: the deactivation of Autorun on machines running variants of the Windows operating system. Unfortunately, the spread of Disk Knight is uncontrolled, and as such has become a nuisance in and of itself.

In order to create an analog without becoming either an additional infection vector or a nuisance, careful consideration was given to several salient elements of the target environment.

2.1 Environment

As noted in [18] and several other studies, Windows, particularly the XP variant, is the overwhelming favorite in the developing world, boosted in particular by the ease of acquisition of pirated copies in various IT shops, bazaars, and markets. As summed up nicely by a quotation in the aforementioned study [18], "Who in today's world uses a genuine copy of Windows Sir?" Windows' ubiquity is also driven by the familiarity of the interface to most casual computer users. A direct consequence of this fact is that the vast majority of virus infections in the wild are on this the Windows x86 platform. As such, Innoculous is designed specifically to address infections on this platform, and is not designed with other operating systems in mind.

Another direct consequence of this fact is that while the hardware capabilities of machines will vary widely (especially network connectivity, memory, and processor speed) most machines modern enough to run Windows XP will share several hardware features. The most significant among these for our purposes are CD/DVD-ROM drives and USB ports.

2.2 Data Logging

As one stated goal of the Innoculous project is to acquire rich data about virus infections, a writable medium is necessary. Several alternatives were considered, including pairing a read-only CD-ROM containing binaries with a USB drive whose sole purpose would be to record logs from the binaries, but this was abandoned as impractical, both because

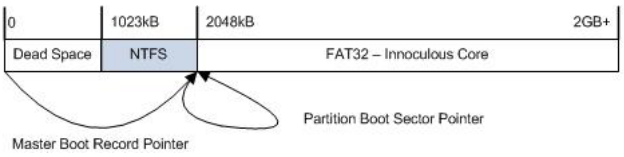

Figure 2: Example partition layout of Innoculous key

the loss of either would render the system incapable of running and because the USB drive, without special measures to prevent it, would itself become a carrier for viral infections.

Ultimately, a single, self-contained USB memory key was selected, but additional effort (explained in 2.4) was necessary in order to ameliorate the infection problem. The existence of the problem is made more than sufficiently clear by the prevalence of USB-transmitted viruses even in some of the most unlikely places, such as brand new USB digital photo frames [5] and even USB keys distributed by IBM at AusCERT, a prominent security conference [10].

2.3 Infection Cleaning

The other primary goal of Innoculous, the cleaning of virus infections, clearly necessitates an antivirus solution. While several free and commercial variants are available, the licensing terms of most of those examined were either draconian or unclear, and very few offered a tool that could be used without installation onto a target machine, which is a risky proposition on a machine that could already be infected. In such a case, files could be infected, corrupted, or otherwise disabled during the installation process.

These considerations led to two important design decisions: First, Innoculous needed a self-contained and preferably sciprtable, command-line driven interface. Second, some measure must be taken in order to prevent disabling of the antivirus engine or corruption of the logs by viruses that might exist on the machine being scanned.

After considering the various commercially available alternatives, Panda Security's antivirus offering was selected as it is explicitly free for use for not-for-profit and research purposes, it runs as a command-line interface, and uses a modern engine with a reasonably highly rated set of definitions [6, 7], a potential improvement over Quick Heal [15], which, though it is well-liked by its users and is the prevalent [18] antivirus in India (a probable first deployment milieu), has fewer professional reviews and mixed results in the reviews that do exist.

2.4 Infection Prevention

Given the design goals and decisions already in place, it remained to prevent the passing of infections via the Innoculous USB key itself. This was accomplished using several nuances of Windows' USB Mass Storage driver and its interaction with USB devices.

Windows variants from Windows 2000 through to Windows 7, including server versions, recognize only the first partition that exists on any USB memory key, and do not themselves have any capability to create multiple partitions on such devices. In observance of this fact, Innoculous is installed on a second partition on a USB stick, after a dummy 1 megabyte NTFS partition (the minimum size), which is presented to Windows. In order to partially mitigate USB threats, this 1 megabyte partition has its entire capacity occupied by a dummy file with a known hash, making the partition tamper evident and proving too small for many infections with large or advanced payloads. In addition, the small size of this partition will discourage users from storing their own personal data on these devices.

While Innoculous thus is inaccessible from the Windows operating system, by installing a Master Boot Record (MBR) on the USB memory key and pointing it to the second partition, as well as a small edit to the partition boot table, this second partition can be booted into. As such, Innoculous was designed to be run from outside Windows.

3. IMPLEMENTATION

Innoculous is implemented using Windows PE 3.1 32 bit, which provides a preinstallation environment based on Windows 7 SP1. This provides a Windows-like environment along with various tools and OS APIs that ease the task of scripting the entire solution to interface with various Windows versions.

3.1 Custom Scripting

The core of Innoculous is a script written in VBScript, which can be supported by a module included during the assembly of the WinPE image. The script is run after WinPE after the kernel boots and all drives and network adapters have mounted, and has the following functionality:

1. Displays the key's hardware ID/serial number
2. Presents the user with an option to replicate a child key (cf. 4.1)
3. Asks the user for the PIN, ZIP or other postal code of their current location, if available.
4. Presents the user with an option to start a scan. If A scan is started:
   (a) Records serial numbers of all hard drives in the system
   (b) Begins scan using Panda Antivirus, storing verbose logs
   (c) Deactivates Autorun using command-line registry editor
   (d) Records salient information about machine including Windows serial number, installed patches, etc.
5. If network connectivity is available:
   (a) Checks for updated virus definitions from a preconfigured IP address
   (b) Compresses and uploads any existing scan logs
   (c) Records system time skew against NTP server

3.2 WinPE

The WinPE image is assembled using the Windows Automated Installation Kit, the aforementioned script, Panda Antivirus binaries and definitions, and several driver packs including network drivers, Windows Management Instrumentation (WMI), and a configuration file.

The AIK then creates a bootable ISO image that serves as the distributable package for Innoculous.

WinPE's limited implementation of the general Windows APIs has both positive and negative aspects for our use. Because rich APIs such as .NET are not available, certain advanced diagnostics and data collection are impossible to do in this environment without substantial reimplementation. However, because many other high-level APIs and core Dynamic Link Libraries (DLLs) are not available, some malware variants, which use these rather than lower-level calls in order to keep their payloads small, fail to function even if they are deliberately run within this environment.

Certain functionality not provided by WinPE is added through third-party software. In particular, in order to mitigate the threat of man-in-the-middle and masquerade attacks during the update phase outlined in 3.1, the image is augmented by Cygwin [12] core libraries and utilities including `scp` and several hash and signing tools. These are used to verify server identities using a known server key, verify incoming updates using a known public key against a private signing key, and used to establish secure ssh tunnels for transmission of logs. GnuPG [16] is used in the prototype to encrypt logs as they are generated, but due to an incompatible license, another option will be used during deployment.

3.3 USB Key Preparation

A USB key at least 2GB in size is necessary for Innoculous to run, though larger keys are obviously preferable. While it is possible to prepare the key using Windows tools, for simplicity I prepared the device on a Linux machine using the following steps:

1. Using `parted`, an NTFS partition is created from 1023kB to 2MiB. This creates a 1 megabyte (1024kB) partition, which is the minimum size supported by any modern filesystem supported by Windows. For contrast, the next smallest is FAT16, which has a minimum partition size of 16 megabytes. It is important to note that while this partition does not begin aligned on a cluster boundary, because the dummy partition is not meant to be frequently read from or written to, the performance impact of this fact is negligible.
2. Using `mkntfs`, the NTFS partition is formatted to NTFS.
3. `parted` is then used to create and format a FAT32 partition comprising the remainder of the device.
4. A Windows PE image, assembled as described in 3.2, is imaged onto the FAT32 partition using `dd` or `partimage`.
5. `install-mbr` or other Master Boot Record program is used to install the MBR onto the USB key and point it to the second partition, e.g.

   `install-mbr -p2 -e2 -v /dev/sdb`.
6. Using the output from `fdisk -ul`, the start boundary is encoded into hexadecimal using, e.g. `printf`, and inserted in little-endian format at position 0x1C of the second partition. This can be done using any hex editor, such as `hexedit` on the device, e.g.

   `hexedit /dev/sdb2`.

It is noteworthy that while the initial preparation of the keys/images is somewhat complex, the keys self replicate as noted in 3.3, limiting this complexity to the initial creation of the key image.

3.4 Deep Forensics

As the Innoculous installation, when run, has access to all files resident on the host machine's drives, it is possible to copy various files from the computer for forensic analysis regarding behavior, such as Windows' Temporary Internet Files folder, browsing history, etc. There are significant privacy and ethics issues surrounding such use, but the laws and customs surrounding these topics differ from region to region.

Access to these data, properly redacted, could prove to be a significant source of insight into infection vectors and browsing habits in the developing world. This functionality, however, is not currently implemented given the murky ethics surrounding the issue of privacy, particularly in a context in which a license agreement or consent form might not be fully understood given limited literacy. Furthermore, as automated redaction is unlikely to be completely effective, some degree of personal identifying information is certain to remain with the corpus of collected files making dissemination for research purposes an ethically risky proposition.

With these factors in mind, Innoculous is unlikely to pursue this avenue, particularly given the decentralized command-and-control structure of these keys.

4. DISTRIBUTION

For Innoculous to effectively gather a large corpus of data, the system must gain adoption across more than just the research community. In the subsequent sections, I outline both technical considerations necessary to enable arbitrary growth as well as an incentive model to encourage use by business and private individuals alike.

4.1 Replication

The script that serves as the core of Innoculous also contains the ability to replicate the entire system to another USB key. It does this using the AIK builder binaries as well as Windows versions of partitioning tools to create a direct copy of itself.

In the process of replication, the parent key records the serial number of the USB device it is replicating itself to. In addition, the replicated key is initialized with the hardware value of its parent, creating a bidirectional link that, as the keys are replicated, creates a graph of keys.

4.2 Incentivization

This graph of keys is critical to the incentivization model, essentially a bounty on new virus types encountered and number of machines scanned. Bounties are paid out on:

- The first n samples of a virus in a given context (i.e. city, town, etc.)
- Each m unique machines scanned as defined by the serial numbers of their hard drives. Using hard drive IDs prevents intentional reinfection and disinfection on the same machine multiple times.

where n, m and the bounties paid on each are defined on a per-context basis. Bounties paid on first samples might be, for example, Rs.8, or approximately USD 0.16.

In order to encourage users of the system to replicate their keys and give them to others, I adopt a system analogous to that which the MIT Red Balloon Challenge Team [14]

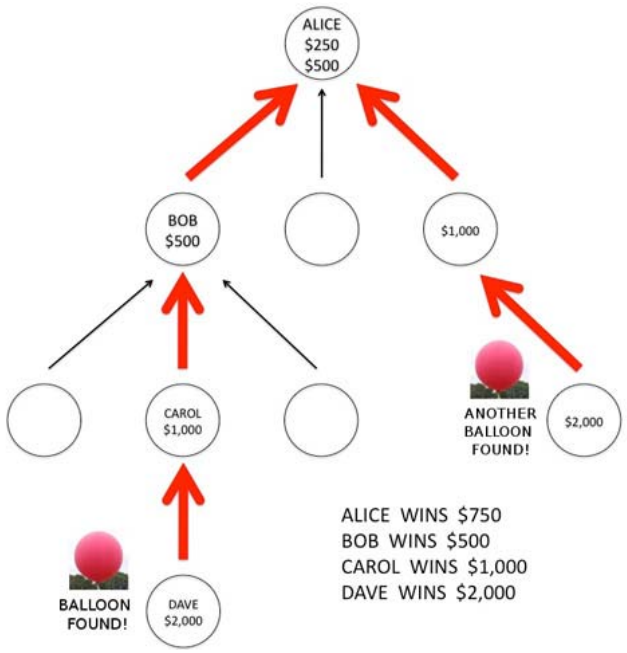

Figure 3: Bounty distribution used by the MIT Red-balloon Challenge Team. Image from [13]

used during the DARPA Network Challenge [9], as illustrated in Figure 3. In this model, bounties would be paid out starting with the finder and then geometrically smaller proportions to the finder's parent, grandparent, etc. Explicitly, $\frac{1}{2}$ would be paid out to the finder, $\frac{1}{4}$ to the parent, $\frac{1}{8}$ to the grandparent, etc: Rs.4, Rs.2, and Rs.1 using the bounty above. While these amounts are small, given the large numbers of infected machines and potentially multiple infections per machine, this could represent a notable revenue stream in the developing world.

The behavior this encourages is to produce as many first-degree links as possible in order to garner proportionally larger amounts of rewards, as well as scanning more machines.

Bounties would be collected either via regional micropayments (e.g. M-PESA), paid out in mobile phone minute equivalents, or picked up in cash at set times at sponsoring organizations.

Optionally, a web-based leaderboard could be implemented to further encourage competition and dissemination. It is also important to note that these bounties are in addition to any monetary compensation that the people with these keys receive for the service of cleaning machines.

No analogous incentive system is known by this author to have been attemped in the developing world, so the acceptance of and response to such a scheme may prove enlightening and interesting in its own right.

4.3 Controls

In order to maintain reins on the system, several controls may be optionally implemented on the keys:

- A usage-based suicide gene that would wipe the key once n scans had been completed and uploaded at some internet-connected machine.
- A time-based suicide gene that would wipe the key at a given date, verified against a known NTP server on some internet-connected machine.
- Generational limits for how many generations from the first tier of keys distributed may be replicated.
- Invoked self-destruct that, when triggered by the server, will cause the key to delete itself upon its next check for virus signature updates.
  - This can be made specific to certain hardware IDs to prune precise parts of the graph
- Invoked disabling of self-replication, forcing any given key to be a leaf node in the graph.

These controls and, indeed, any combination of them can be used to allow or restrict expansion of the Innoculous network to any specified degree. Additionally, as geolocation data about IPs used to dial home for updates is recorded, it is possible, to some degree, to restrict geographic expansion as well, by simply causing any key that communicates using an IP address outside a particular geographical area to deactivate.

5. ANALYSIS

The data Innoculous returns can be used for various analyses that would provide insight into the state of virus infection and end-user computing in the regions in which it is deployed. Some simple examples of these include:

- Geographic spread analysis: Similar to the same analysis in traditional epidemiology, this could illustrate the spread levels and densities of particular strains of viruses, and would prove especially illuminating with respect to viruses that use USB keys and other storage devices as vectors, rather than internet-based malware and email attachments. Also of interest would be the correlation between the socioeconomic statuses of particular areas and the types of viruses detected.
- Strain analysis: Based on the 'birthday' of each strain of virus, worm, or other malware, it is possible to determine certain data regarding the age, spread rate, and infection vector of observed viruses.
- Reinfection: As some machines will likely be scanned more than once given a sufficiently large network of Innoculous keys, data will emerge regarding subsequent reinfection of machines that have been cleaned before.
- Piracy analysis: Determining what proportion of Windows installations are genuine and which may have come infected with viruses.

6. RELATED WORK

While a great deal of literature exists regarding viruses and security methodologies on the cutting edge of technology, precious little of this is salient to the developing world context. Additionally, nearly all the data available surrounding the issue of viruses in these contexts is purely anecdotal and provides little in the way of deep analysis into prevalence, ecology, geographical spread, or real-world economic impact.

The nearest to the on-the-ground infection data Innoculous would provide is the malware traces collected by Johnson et al. [22], but these traces only capture malware traffic on networked machines and provide no information about offline machines.

As far as I am aware, this is the first proposal of an instrumented device to clean and log viruses.

7. CURRENT STATE AND FUTURE WORK

Innoculous is currently in a state of active development, and core scanning and logging functionality have been successfully implemented and tested against infected machines in a lab environment. Field testing is scheduled to begin in summer 2011, initially on a limited scale in Bangalore, India.

As Innoculous spreads, the data that it returns should allow for direct improvements in the system to combat the particular types of infections seen. Moreover, the corpus gleaned should provide insight into various new research directions in the particular challenges posed by security in the developing world as well as practical methods to block infection vectors particular to the space.

8. CONCLUSIONS

In this paper I have presented Innoculous, intended to be a system for robust data collection for virus infections in the developing world. The system is designed around the particular environment in question, and not only directly combats the problem of viruses, but motivates others to participate in doing so.

The incentivization model uses both a native desire to work without the destructive overhead that viruses impose as well as direct financial rewards to encourage the spread and use of the Innoculous system, which is designed to self-replicate in a controlled manner.

Within the design of Innoculous is both a novel defense against infection of the host USB device as well as novel methodologies for offline data collection and reporting.

While a limitation exists in the system around the cost of the physical USB keys necessary for the system to replicate, I expect the incentives will partially offset this cost, which is itself decreasing rapidly over time.

The use of the Innoculous system, if widespread, will provide the research community with a detailed corpus of data regarding virus infection rates and types at low cost while simultaneously providing revenue streams for small business owners and individuals in the developing world and raising awareness of the problems presented by virus infection. As a bonus, it also will provide a social network graph of people in the region(s) in question who are likely to be considered local computer power users, information that could help establish a valuable social network in deploying future projects.

9. ACKNOWLEDGMENTS

I would like to thank Bill Thies for his invaluable help in developing some of these ideas, as well as various friends and colleagues at NYU and Microsoft Research India.

I would also like to thank Panda Security for providing their command-line scanner free for use in research and not-for-profit projects.

10. REFERENCES

[1] Kaspersky labs. `http://www.kaspersky.com`.

[2] McAfee. `http://www.mcafee.com`.

[3] McAfee Security - World Virus Map. `http://mastdb3.mcafee.com/VirusMap3.asp?name=VirusMap`.

[4] Disk Knight is 'beneficial' virus. `http://it.toolbox.com/blogs/adventuresinsecurity/disk-knight-is-veneficial-virus-18456`, 2007.

[5] Best Buy issues security warning on Insignia digital picture frames. `http://news.cnet.com/8301-10784_3-9857364-7.html`, 2008.

[6] Panda Internet Security 2010 review. `http://www.techradar.com/reviews/pc-mac/software/utilities/anti-malware-software/panda-internet-security-2010-637250/review`, December 10, 2009.

[7] Review: Panda Internet Security 2010 - Security. `http://www.thetechherald.com/article.php/200926/3940/review-panda-internet-security-2010`, June 25, 2009.

[8] Trojan found in pirated Apple iWork software. `http://news.cnet.com/8301-1009_3-10148359-83.html`, January 22, 2009.

[9] Darpa Network Challenge. `https://networkchallenge.darpa.mil`, 2010.

[10] IBM unleashes virus on AusCERT delegates. `http://www.itnews.com.au/News/175451,ibm-unleashes-virus-on-auscert-delegates.aspx`, 2010.

[11] Seventh Annual BSA/IDC Global Software Piracy Study. `http://portal.bsa.org/globalpiracy2009/studies/globalpiracystudy2009.pdf`, May 2010.

[12] Cygwin. `http://www.cygwin.com`, 2011.

[13] How It Works. `http://balloon.media.mit.edu/media/images/how_it_works.jpg`, 2011.

[14] MIT Red Balloon Challenge Team. `http://balloon.media.mit.edu`, 2011.

[15] Quick Heal. `http://www.quickheal.com`, 2011.

[16] The GNU Privacy Guard - GnuPG.org. `http://www.gnupg.org`, 2011.

[17] E. Adomi. Overnight Internet Browsing Among Cyber Café Users in Abraka, Nigeria. *Journal of Community Informatics*, 3(2), 2007.

[18] P. Bhattacharya and W. Thies. Computer Viruses in Urban Indian Telecenters: Characterizing an Unsolved Problem [Under Submission]. 2011.

[19] E. Brewer, M. Demmer, M. Ho, R. Honicky, J. Pal, M. Plauche, and S. Surana. The Challenges of Technology Research for Developing Regions. *IEEE Pervasive Computing*, 5(2), April 2006.

[20] A. Garuba. Computer Virus Phenomena in Cybercafé. *Security and Software for Cybercafes*, 2008.

[21] A. Haseloff. Cybercafes and their Potential as Community Development Tools in India. *Jounral of Community Informatics*, 1(3), 2005.

[22] D. Johnson, V. Pejovic, E. Belding, and G. van Stam. Traffic Characterization and Internet Usage in Rural Africa. In *WWW 2011*, Hyderabad, India, 2011.

[23] K. Kinyanjui. Kenya tops list of EA countries worst-hit by computer viruses. *Business Daily Africa*, September 9, 2010.

[24] C. Michael. Computer viruses slow African expansion. *The Guardian*, August 13, 2009.

[25] S. Wyche, T. Smyth, M. Chetty, P. Aoki, and R. Gringer. Deliberate interactions: characterizing technology use in Nairobi, Kenya. In *CHI 2010*, Atlanta, GA, 2010.