Automated Computer Forensics Training in a Virtualized Environment

By

Stephen Brueckner, Frank Adelstein, David Guaspari, Joseph Weeks

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2008 USA

Baltimore, MD (Aug 11ᵗʰ - 13ᵗʰ)

Automated computer forensics training in a virtualized environment

Stephen Brueckner[a,*], David Guaspari[a], Frank Adelstein[a], Joseph Weeks[b]

[a]ATC-NY, United States
[b]Air Force Research Laboratory, United States

A B S T R A C T

Keywords:
Digital forensic training
Computer training
Virtualized training
Automated assessment
Automated evaluation

The CYber DEfenSe Trainer (CYDEST) is a virtualized training platform for network defense and computer forensics. It uses virtual machines to provide tactical level exercises for personnel such as network administrators, first responders, and digital forensics investigators. CYDEST incorporates a number of features to reduce instructor workload and to improve training realism, including: (1) automated assessment of trainee performance, (2) automated attacks that respond dynamically to the student's actions, (3) a full fidelity training environment, (4) an unrestricted user interface incorporating real tools, and (5) continuous, remote accessibility via the Web.

© 2008 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.

1. Introduction

The need for training in digital forensics is great and growing. Automated support for training has typically been limited to the examination of disk images and to the mastery of specific forensic tools.

ATC-NY has developed CYDEST to provide a rich and flexible environment for cyber defense training. Its principal features are

- support for highly realistic, "immersive" training scenarios (including ongoing attacks and live forensics),
- automated evaluation of students' performance.

CYDEST obtains a high degree of realism by *virtualizing*, rather than simulating, the networks and systems that a student is tasked to defend or investigate. Such an infrastructure can support arbitrarily rich training scenarios. Fig. 1 shows a typical example of a training network. The virtualized networks, run on a few physical hosts, are indistinguishable from a heterogeneous collection of networked servers, desktops, and infrastructure running a variety of operating systems, each with its own installed base of application programs, administrative tools, etc.

This paper briefly describes the architecture of CYDEST, its current capabilities and future milestones, a description of an example training scenario, and an experiment in which 18 students used CYDEST to conduct a laboratory exercise in a university forensics course.

2. Overview of CYDEST

2.1. Virtualization

CYDEST implements virtualized hosts with Xen, an open source *hypervisor* (or *virtual machine monitor*). Xen runs directly on the hardware as an operating system control program. One or more *guest* operating systems may be run as virtual machines on top of Xen. One of the guests has special privileges,
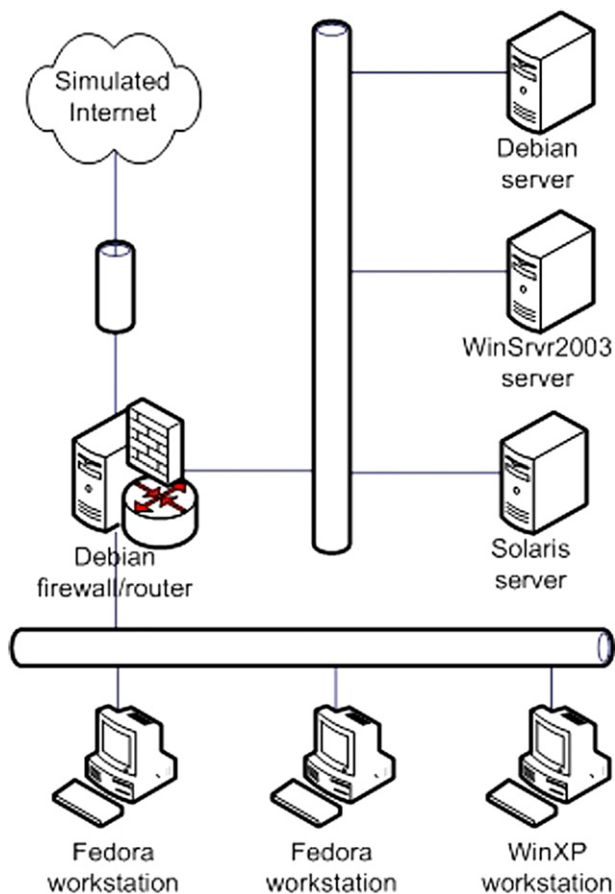
Fig. 1 – Heterogeneous virtualized training network.

Fig. 2 – CYDEST architecture.

including direct access to the physical hardware and the ability to monitor and control the other guests. CYDEST uses a modified Linux OS to serve as this privileged guest.

The hosts visible to a CYDEST student are *unprivileged* guests. New hardware architectures (Intel's VT-x, AMD's AMD-V) have been created to allow the virtualization of unmodified operating systems running as unprivileged guests. Xen 3.0 can exploit those architectures and thereby allow CYDEST networks to contain hosts running proprietary operating systems such as Windows.

2.2. The architecture of CYDEST

The architecture of CYDEST is illustrated in Fig. 2. Its three main components are the target network, the attack network, and the control network – networks of *virtual machines* (VMs) running on a small number of physical hosts. The dotted line represents the network that carries traffic internal to the exercise and therefore visible to the trainee. The solid line represents a separate, isolated "out-of-band" network that carries traffic for monitoring and controlling the exercises, and is not visible to the trainee.

The *target network* contains the assets that the trainee is responsible for defending or examining. The offensive *attack network* generates external traffic – both hostile traffic and benign "background noise" – directed at the target network. The attack network is often used in scenarios to simulate
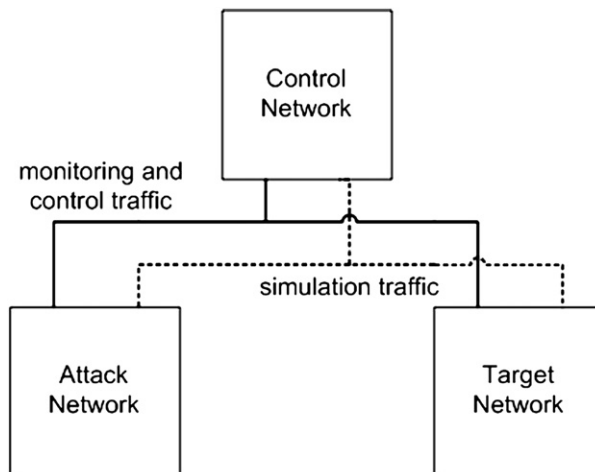
the Internet. (In the case of insider attacks the hostile traffic comes from within the target network itself.) These two networks are configured specifically for each scenario. The *control network* hosts permanent infrastructure that is not scenario-dependent. It includes the *gateway*, which allows remote user login and mediates inter-network scenario traffic, the *attack generator*, which carries out (possibly dynamic) attacks, and the *trainee evaluator*, which provides automated monitoring and assessment.

CYDEST's rich, virtualized environment comes at a price, for monitoring and controlling its elements presents some difficulties. Individual components are complex, active entities whose states can be manipulated only indirectly and cannot be known without probing. The control network tracks the states of components with out-of-band monitor processes invisible to the student. However, only state deemed relevant to a training exercise is selected for monitoring, so the resulting database of state information contains a very small subset of the true state of the virtual machines. Finally, the database of state information is often a few seconds out of date, but that causes no problems, because scenarios unfold at a human pace.

A virtual host may unexpectedly crash; and if that happens CYDEST reboots it using one of two specifiable recovery modes – restoring either to a "latest" configuration (and preserving history information, so far as possible) or to some predefined baseline configuration.

One final complexity: exercises are not literally repeatable. For example, variations in timing mean that a student who repeats an exercise will not reproduce the identical sequence of states, log entries, etc. We have devoted considerable effort to reducing these variations to a level that will be pedagogically unimportant.

2.3. Assessment

A CYDEST exercise is "full fidelity": the virtualized components are indistinguishable from "real" components, and the student is free to probe them as desired (though typically limited to using some specified collection of tools). This

freedom and complexity pose a number of problems for assessment, for we wish to check not only that a student got the right answers, but that he *got them in the right way* (or, at least, in a reasonable way) – lucky guesses shouldn't count. CYDEST's assessment functions use both active and passive observation.

Active observing monitors a student's actions and both the direct and indirect results of those actions. For example, CYDEST can observe the student action of reading a certain file (using a tool from some specified list: less, more, vi, emacs, etc.). However, if the student views the file using another, unanticipated method, CYDEST can observe the direct result of reading the file (by checking the file's access time). The first observation method provides more detailed information for the auto-assessment process, and the second observation method provides more assurance of capturing the required information.

CYDEST provides the basic infrastructure for monitoring, but the scenario designer must specify which things to observe and how to interpret those observations. The scenario designer must decide whether reading the file is a sufficient proxy for having read it and extracted a key piece of data. If not, we can require the student to report the key data in his notebook (see below) or monitor some other proxy (such as whether that key piece of data is used as an argument to some operation invoked on the command line).

Passive observing relies on reports from the student – either responses to direct queries (e.g., Who leaked the data?) or entries in a *notebook* for recording structured information about the course of an investigation. To make entries in the notebook the student navigates through a simple shallow tree in CYDEST's notebook window to reach a leaf that pops up a form for entering details. For example, a student may choose ''log files'' from the top level menu, then the sub-entry ''system logs'' – at which point CYDEST displays a form for entering system name, log name, and the relevant log entry itself (which can be supplied by cutting and pasting).

Each notebook form also contains a text box for unstructured comments, including conclusions drawn from the observation. CYDEST passes those comments along to an instructor when it creates a summary log of each session.

The requirement to keep a notebook is not artificial, since standard guidelines on incident response and digital forensic investigation (e.g., Federal Communications Commission, 2001; National Institute of Standards and Technology, 2004; United States National Institute, 2001) recommend that investigators document their work with pen and paper.

Actively collected state information and passive notebook entries can include booleans, numerics, or alphanumeric strings. The scenario designer determines a priori the correct (or incorrect) values, numerical thresholds, or regular expressions (respectively) for each relevant state or notebook parameter, along with an appropriately weighted integer score. An evaluation function aggregates these individual scores into a final grade for the student, subject to review of session audit logs (see Section 2.4.5).

2.4. A CYDEST session

CYDEST has a Web-based interface, and can be installed locally or accessed over the Internet. Client requirements are light: only a Java-enabled Web browser to access and interface with the CYDEST server. The top level of the interface provides an online reference manual, an extensible knowledge base of general information (about networking, forensics, etc.), and a screen from which the student may choose a scenario to run. (The instructor's interface provides additional functions, such as the ability to observe the log of events that the control network monitors.)

Once a student has chosen a scenario, CYDEST displays the target network for that exercise, together with controls making it possible to power individual VMs on or off, to reboot them, and to start and stop the exercise. The student can click on a VM icon to access it via a remote desktop connection, which pops up in a separate browser tab. The remote desktop is implemented using platform-agnostic virtual network computing (VNC) software served as a Java applet. The notebook interface is available under a separate tab.

CYDEST scenarios can include both ongoing attacks (denial of service, worms, etc.) and after-the-fact forensic investigation of live systems. This section describes a scenario that is purely forensic.

2.4.1. The mission
The student is responsible for the network used by the shipping department of a nationwide retail chain. This target network, shown in Fig. 3, is spread over two physical hosts. It consists of a *servers* subnet and a *staff* subnet. The servers subnet contains
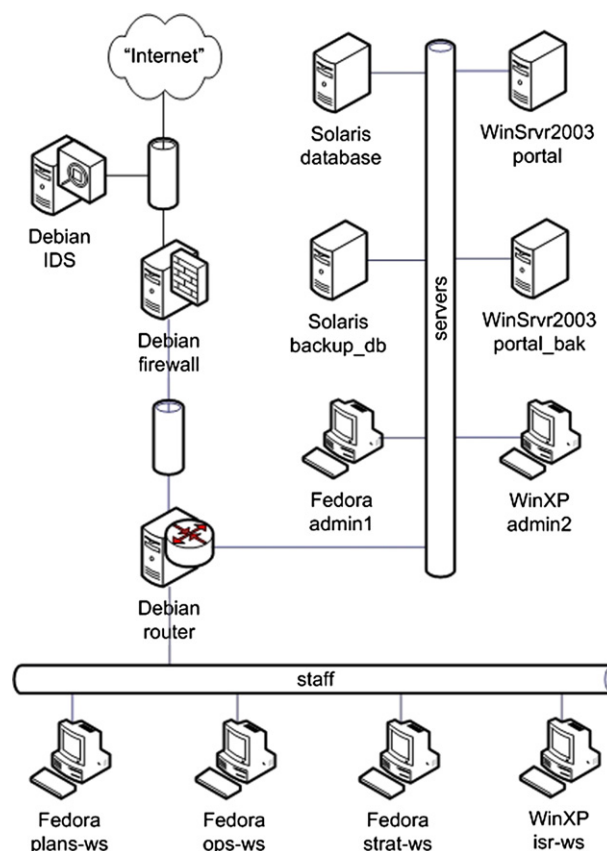

Fig. 3 – Forensic scenario target network.

- sensitive information (a database and its backup) hosted on two Solaris VMs,
- portal services (e.g., access control, email, Web, FTP, DNS) hosted on two Windows Server 2003 VMs,
- two admin workstations the trainee can log into,[1] one running Linux Fedora and one running Windows XP.

The staff subnet contains four workstations (Linux and Windows) from which workers access the databases and services in the servers subnet. Each of these workstations has one primary user (whose login account name is given in parentheses, below):

- Plans *(iago)*: inventories, ordering.
- Operations *(regan)*: dispatchers, dynamic replanning.
- Strategy *(helen)*: sales trends, climate prediction.
- ISR *(bianca)*: weather hazards, road closings, ordering problems.

Three other VMs provide network infrastructure – an intrusion detection system, a firewall, and a router (all running Debian Linux).

CYDEST uses Instant Messaging (IM) to deliver communications from other employees and instructions and pedagogical hints from the control network. The exercise proper begins when the student receives an IM stating that company trucks are being robbed, possibly with the aid of inside information; and further, since the route for the most recently robbed truck was changed at the last minute, there is a small window of time in which the most recent leak, if there was one, must have occurred.

The student's task is to locate and record evidence, follow the evidence trail, identify the attacker, and exonerate innocent users. The forensic tools to be used include standard Linux command line utilities and the Online Digital Forensic Suite™ (ATC-NY), developed at ATC-NY.[2] OnLineDFS™ allows an investigator to access hosts remotely, gather data from them, and display and search that data for evidence. For present purposes, it represents an instance of our ability to integrate the use of proprietary tools into CYDEST scenarios.

2.4.2. The attack
The attack was mounted by an unsophisticated insider (login name *regan*) attempting to conceal her actions by using two other accounts (*helen, iago*). Within the time window indicated by the IM, *regan* performs the following actions:

2:05:23 Logs into *helen's* strategy workstation using her own credentials.
2:06:16 Tries four times to log in to mysql on db_backup as *helen* but her password guesses fail.
2:10:26 Logs in as *regan* on db_backup.

2:10:29 Performs a series of queries to determine location and plate number of a truck, saving the results in file called `.orders.swp`.
2:19:00 Copies the script `.mailer.pl` from Operations to Strategy. Attempts to send email as *helen* with `.mailer.pl`, but fails because sendmail isn't active on *helen's* machine. Sends `.orders.swp` from Strategy to Operations and deletes `.orders.swp` and `.mailer.pl` from Strategy. Logs out of Strategy.
2:29:23 Logs into Operations using regan's own credentials and checks `.orders.swp` with vi. Sends email as *iago* with `.mailer.pl`. Deletes `.orders.swp` from a separate terminal while it's still open in vi.
2:32:00 Logs out of Operations.

2.4.3. The evidence
The attacker leaves a forensic trail consisting of two files on Operations – the stolen data file `.orders.swp` and the mailer script `.mailer.pl` – along with entries in six standard log files:

- bash history files for *regan* on Operations and Strategy,
- the accounts login log and *iago* email contents log on Portal,
- *regan's* mysql client log on Strategy,
- the mysql server log on db_backup.

2.4.4. Assessment
The student should identify *regan* as the leaker – and should also exonerate *helen*, whose workstation *regan* logged onto, and *iago*, whose email account was used to transmit the stolen information.

Active observation. In this scenario the control network monitors the bash commands executed in the trainee's command line terminal and the logs generated by OnLineDFS™. It also keeps track of which workstations the student logs into. These observations allow CYDEST to determine whether the student is examining relevant files and relevant workstations. Points are awarded for each evidence file accessed – more for files on the "investigative pathway" (i.e., the trail of evidence that follows from the initial clues) and fewer for ancillary files that support the investigation but are not critical to it. Points awarded may be increased or decreased based on the amount of time elapsed before the student accessed relevant files.

Logging onto a workstation that is not on the investigative pathway results in deductions (and will result in hints sent via IM).

Passive observation. The student is required to record evidence in the notebook as it is discovered. CYDEST can automatically assess the validity of notebook observations about specific entries found in the log files. When the student completes the investigation, CYDEST sends a series of IMs querying the student – asking for the name of the attacker, the channel used to exfiltrate the sensitive data, etc.

2.4.5. The CYDEST log
CYDEST also maintains a session log that records the occurrences of all the events it has been told to monitor. These logs serve several purposes: students may review past

---

[1] The student gains access to the network by logging into a virtual machine configured to allow VPN connections.
[2] OnLineDFS originated as an SBIR sponsored by AFRL in Rome, NY. It is marketed and maintained by Cyber Security Technologies.

sessions. Instructors may examine logs for insight into students' mistakes, and to correct the grading of an exercise (if, e.g., the evaluator has responded inappropriately to some novel or unexpected actions of the student). Scenario developers may use the logs to help determine the effectiveness of scenarios and evaluation parameters.

3. Practical applications

We conducted a field test of CYDEST using the scenario discussed in Section 2.4. Eighteen undergraduate students enrolled in the winter 2008 forensics course at the University of New Orleans were assigned to run the exercise. Seven teams of 2–3 students each were given two 2-h time slots to complete the assignment.

Teams completed the exercise in varying amounts of time; approximately half of the teams used both of their time slots, and half only used the first. Teams found from 4 to 7 of the 8 evidence files and all were successful in determining the inside attacker and exonerating at least one other user. No team grasped all of the scenario's nuances, but given time constraints all performed satisfactorily. Although students were instructed to make notebook entries for each piece of evidence found, on average teams noted only 60% of the evidence they examined.

Students filled out questionnaires providing feedback on both the CYDEST framework and the scenario. Feedback on CYDEST included notes on application performance and ideas for improving the user interface, particularly the notebook and the IM system. Feedback on the scenario included some inconsistencies in timestamps and footprints left from scenario setup. We note that setting up realistic, high-quality forensic scenarios is a difficult problem to which general solutions are needed.

We benefited greatly from the field test in identifying both general areas for improvement and specific bugs and configuration errors. We will use the feedback and performance results to improve the CYDEST framework and both the current and future training scenarios.

CYDEST was initially developed to train the operators of military computer networks. We are currently demonstrating its use as a tool for teaching forensics in an academic setting. Future plans are to use CYDEST to train defenders and investigators in a commercial, as well as military, environment.

4. Related work

4.1. Forensics training courses

As discussed in the remainder of this section, digital forensics training is primarily conducted in the classroom or with traditional online courseware. We are unaware of any frameworks similar to CYDEST, i.e., that provide automated training and assessment using detailed live scenarios. There are some related efforts in the realm of network security, but most of these are part of programs sponsored by the military and are not available to the general public.

Many educational institutions (see, e.g., the listing in http://www.e-evidence.info/education.html) and training firms offer courses in forensic investigation. Training firms typically prepare a student to take the test for a particular certification, such as Certified Computer Examiner (from the International Society of Forensic Computer Examiners). The offerings at educational institutions are typically broader.

The SANS Institute (http://www.sans.org/training/description.php?tid=411) currently offers an advanced forensics course on analyzing and recovering data seized from live systems. Exercises use the VMware hypervisor to run several different operating systems simultaneously on the same workstation.

ATC-NY has created a forensics course for the Virtual Training Environment of CERT (https://www.vte.cert.org/vteweb/about.aspx#courselist). The course trains students in the use of OnLineDFS™, and includes written course notes, a video lecture, and a guided live forensics exercise in which they collect volatile data from a running network.

4.2. Pedagogical theory

Constructivist learning theory (Dewey, 1927; Kirschner et al., 2006; Piaget, 1950; Vygotsky, 1978) proposes that learners are not passive recipients, but actively construct the knowledge that they acquire, and that the more realistic and authentic the learning environment, the more likely it is that learning will transfer to the work environment. It does not advocate a single pedagogy; but rather any approach that promotes "hands-on" and "personalized" learning – which the National Academy of Engineers has identified as a major challenge for the future (National Academy of Engineering of the National Academies, 2008).

One-on-one tutoring is the most effective way to provide personalized learning, but is impractical when limited manpower rules out sufficiently low student-to-instructor ratios. CYDEST supports personalized learning by automating much of the assessment and feedback process, reducing the burden on instructors. Most importantly, CYDEST provides an authentic, realistic learning environment for dynamic tactical exercises, which increases the chances that what is learned in the classroom will transfer to real life forensic challenges.

5. Future work

5.1. Defining scenarios

Thus far, development of CYDEST has created the infrastructure necessary to run and evaluate scenarios on virtualized networks. The next critical task is to provide an efficient way to create scenarios.

The target and attack networks for any scenario can easily be configured with Virtual Network Builder (VNB) – a tool developed at ATC-NY as part of EXAMIN (the EXploit And Malware Incubator) (Brueckner et al.). The graphical interface of VNB allows one to straightforwardly design, configure, and build a virtual network of virtual machines. VNB can currently

configure and provision Linux VMs; future work will automate the process of building Windows VMs as well.

A CYDEST scenario is currently defined by a set of rules: the top level algorithm is a loop that repeatedly selects an eligible rule (a highest priority rule whose *preconditions* are all satisfied) and executes it. The effect of executing a rule may be arbitrarily complex – e.g., it may cause the control network to open up a remote shell on some attack machine and, from that shell, launch a pre-installed attack script; or it may supply a hint (and deduct points) because observations suggest that the student is aimlessly thrashing.

CYDEST stores the state of the scenario and a log of all observed events in a MySQL database and evaluates the preconditions of a rule by querying that database. Queries may refer to the internal states of any of the hosts and switches, information that is regularly refreshed by a monitor process that probes them. The database for any exercise is, by modern standards, quite small and handles queries speedily.

The rule mechanism is low level and unstructured. We are developing a higher-level structured language for specifying scenarios, using a modest variant of the well-known State-Charts notation (Harel, 1987). We currently plan to exploit the infrastructure we have already developed by compiling that structured language into rules.

5.2. Improved assessment

As noted, assessment must determine not only whether the student got the right answers but whether those answers were arrived at in the right way. The student conducting an investigation is making a mental model of a situation, either ongoing or in the past. That model is formed principally by making observations (data from logs, packet sniffers, etc.) and drawing conclusions. The discussion in this section, which could be made more general, will suppose that all observations and conclusions of interest are structured entries in the student's notebook.

The evaluator should judge both the correctness of each observation (e.g., is some information allegedly derived from a log file really there?) and its reasonableness (Is there a good reason to have made that particular observation?). The evaluator should also judge the reasonableness of any conclusion, based on the observations and conclusions recorded thus far.

To make judgments of reasonableness we will use a Bayesian network model of an expert investigator (Pearl, 1988). The reasonableness of a conclusion is measured by its *conditional probability*, given the observations and conclusions so far recorded. The reasonableness of making an observation is measured by its *value of information* – by how much additional information that observation is likely to supply.

Constructing a good Bayesian net is a substantial project. Our initial draft is based on standard guidelines about incident response (Federal Communications Commission, 2001; National Institute of Standards and Technology, 2004).

5.3. Intelligent tutoring

Learning theory proposes that the instructor's role is to facilitate learning by assessing, coaching, and consulting. Assessment is a two-way process between instructor and learner to identify current level of performance and ways to improve. This suggests several long-term goals for CYDEST:

- Development of expert models for trainee evaluation.
- Development of pedagogical agents for comparing trainee performance with expert models and providing prescriptive and proscriptive feedback.
- Automated, intelligent learning management.

6. Conclusions

CYDEST uses virtualization to faithfully model arbitrary, heterogeneous networks and arbitrarily complex attacks upon them. The CYDEST prototype resolves the technical problem of controlling these rich models within automated training scenarios. It provides the infrastructure for defining scenarios and the active and passive monitoring required to perform automated assessment of students' performances. CYDEST's realism and automation, combined with continuous availability over the Internet, provides essential training while reducing instructor workload.

Acknowledgment

REFERENCES

ATC-NY. OnLine digital forensic suite™. http://www.atc-nycorp.com/Products.html.

Brueckner S, Durett G, Inoue H. EXAMIN: tools for malware incubation. In: 23rd annual computer security applications conference, December 10–14, Miami Beach, FL. http://www.acsac.org/2007/wip/WiP%20Inoue%20ACSAC07.pdf.

Dewey J. The public and its problems. New York: Henry Holt & Co.; 1927. p. 126.

Federal Communications Commission. FCC computer security incident response guide; December 2001.

Harel D. Statecharts: a visual formalism for complex systems. The Science of Computer Programming 1987;8:231–74.

Kirschner PA, Sweller J, Clark RE. Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. Educational Psychologist 2006;41(2): 75–86.

National Academy of Engineering of the National Academies. Grand challenges for engineering. http://www.engineeringchallenges.org/; 2008 [accessed 04.03.08].

National Institute of Standards and Technology. Computer security incident handling guide. NIST special publication 800-61; January 2004.

Pearl J. Probabilistic reasoning in intelligent systems. Morgan Kaufmann; 1988.

Piaget J. The psychology of intelligence. New York: Routledge; 1950.

United States National Institute of Justice Technical Working Group for Electronic Crime Scene Investigation. Electronic crime scene investigation: a guide for first responders; July 2001.

University of Cambridge Computer Laboratory. The Xen virtual machine monitor. <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>.

Vygotsky LS. Mind and society: the development of higher mental processes. Cambridge, MA: Harvard University Press; 1978.

Stephen Brueckner is a Senior Principal Scientist at ATC-NY in Ithaca, New York, where he has conducted R&D efforts in the fields of computer security, virtualization, training, secure collaboration, and P2P networking. Formerly a geologist, in 2004 he received his Master's degree in computer science from Oregon's Portland State University, with a concentration in computer security.

David Guaspari, Ph.D. in mathematics, University of Cambridge (1974) is a Staff Scientist at ATC-NY. His research interests include logic, formal methods, programming language semantics, and security. He is a member of the ACM, IEEE, and Wolf's Mouth Theatre Collective.

Dr. Frank Adelstein is the Technical Director of Computer Security at ATC-NY, and provides oversight and guidance to projects at ATC-NY relating to computer security. His areas of expertise include digital forensics, intrusion detection, networking, and wireless systems. He has co-authored a book on mobile and pervasive computing. He received his GIAC Certified Forensic Analyst certification in 2004. A recent research focus is in the area of live forensics. He was the Principal Investigator on projects that resulted in the OnLine Digital Forensic Suite (TM), a live forensics tool, and P2P Marshal, a file sharing analysis tool. He is the vice-chair of the Digital Forensics Research Workshop.

Dr. Joseph Weeks is Chief of Cyber Distributed Mission Operations, Air Force Research Laboratory, Human Effectiveness Directorate at Mesa, Arizona. He is responsible for administrative, financial, and technical oversight of exploratory research and advanced technology development for cyber warfare training. His research interests include learning, performance measurement, and evaluation of computer-based, simulation training. He holds a master of science from Trinity University and a doctor of philosophy from the University of Texas at Austin.