Testing Cyber Security with Simulated Humans

Jim Blythe and **Aaron Botello** and **Joseph Sutton** and **David Mazzocco**
and **Jerry Lin** and **Marc Spraragen** and **Mike Zyda**
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292

Abstract

Human error is one of the most common causes of vulnerability in a secure system. However it is often overlooked when these systems are tested, partly because human tests are costly and very hard to repeat. We have developed a community of agents that test secure systems by running standard windows software while performing collaborative group tasks, mimicking more realistic patterns of communication and traffic, as well as human fatigue and errors. This system is being deployed on a large cyber testing range. One key attribute of humans is flexibility of response in order to achieve their goals when unexpected events occur. Our agents use reactive planning within a BDI architecture to flexibly replan if needed. Since the agents are goal-oriented, we are able to measure the impact of cyber attacks on mission accomplishment, a more salient measure of protection than raw penetration. We show experimentally how the agent teams can be resilient under attacks that are partly successful, and also how an organizational structure can lead to emergent properties of the traffic in the network.

Introduction

As attacks on computer systems grow more common and more sophisticated, rigorous testing is needed to measure the effectiveness of tools that protect against or mitigate the effects of these attacks. Human error is widely recognized as one of the most important sources of vulnerability in a secure system. In a survey taken in 2006, approximately 60% of security breaches were attributed to human error by security managers (Crawford 2006; Cranor 2008). Humans often ignore or misunderstand warnings, underestimate danger, and download infected files or simply disable security mechanisms because of their slowness or complexity (Whitten and Tygar 1999). Even a computer that is switched off or disconnected from the network may not be secure: a social engineering attack may exploit human error by convincing someone to plug it back in (Mitnick 2002).

Frailties are only one aspect of human behavior that impacts our understanding of security. Compared with software systems, humans are flexible and resourceful problem solvers, able to find alternate ways to accomplish their tasks despite failures of resources or services. Different people often perform the same task in different ways, providing a diversification defense from some attacks.

Despite this, human factors are often overlooked when secure systems are tested. One reason for this is the difficulty and cost of creating repeatable experiments that introduce human error into large-scale networks. Human subjects are costly and often require training. Large numbers of subjects may be required for statistical significance and new groups of subjects are needed for repeat experiments due to the learning effect.

We describe an implemented agent system to model human behavior in networks for more accurate security testing without human subjects. Our agents capture some aspects of human behavior that have a significant impact on network traffic. First, they operate standard windows software, using an interface built on VNC by Skaion Corporation. Second, they work in teams to perform a collaborative task, producing task-oriented communication and other traffic with a structure that is not captured by typical statistical traffic generators. Third, they use reactive planning in a BDI framework (Bratman 1987; Morley and Myers 2004) to alter their behavior in response to changes in their environment and respond more flexibly to surprises than is possible in a completely scripted system. Fourth, they include simple models of human physiology and emotion to capture aspects such as degradation in performance due to fatigue or stress.

To our knowledge this is the first agent framework to incorporate these elements to test security systems automatically. In addition, when the agents work towards a team goal we can measure the effectiveness of an attack and of the defenses in terms of changes in the degree of goal satisfaction. This is usually a more important measure than the level of penetration from the attack or the number of resources compromised, but it cannot be measured as effectively using standard approaches with statistical traffic generation and without agent behavior.

In the next section we introduce a scenario based on an insider attack and cloud computing and define a small set of agent types that are combined to provide task-oriented group behavior in the scenario. The following section describes our infrastructure that allows large multi-agent simulations to control desktops on a cyber testing range with

hundreds of host machines and logs agent behavior for subsequent analysis. Next we describe two experiments. The first demonstrates the impact on network traffic and team goal accomplishment of our agents' ability to plan flexibly in changing situations. The second demonstrates emerging complex behavior in the network as a result of the simple behaviors of the agents and their organizational structure. We are continually working to improve this platform with better models of emotion and bounded rationality, richer agent types and standard organization structures. We describe future work and discuss some implications of this work after the evaluation.

Related work

We draw from previous work in human factors in security, agent based simulation and security testing.

Dourish and Redmiles (Dourish and Redmiles 2002) introduce the concept of "effective security" as a more realistic measure of the security of a system than a formal evaluation of the security mechanisms installed. The level of effective security is almost always below the level of theoretical security that is technically feasible in a system, largely due to human error. On the other hand, effective security must be measured end-to-end, taking into account the entirety of the system and the purpose it solves. Thus a high level of theoretical security may be both expensive and unnecessary.

Cranor (08) proposes a framework for reasoning about the security of systems with humans in the loop. She models the human as an information processor based on the warnings science literature (Wogalter 2005). However, this model only captures the human response to warning messages and ignores many important aspects of human behavior, such as the task being performed, collaboration that leads to structured communication, and stress, emotions and tiredness that will affect a human's propensity to make errors. Cranor's approach allows a checklist-style evaluation of a security system. Blythe et al. (11) model human activity to detect when a user would benefit most from a security warning and draw on work in mental models to design effective video-based warnings (Blythe, Camp, and Garg 2011).

Current state of the art traffic generation systems deployed in most security testing either replay recorded traffic or use statistical generators (Wright, Monrose, and Masson 2004). These approaches lack the notion of the human-in-the-loop as well as robustness in the face of failure or response to changes in the environment.

Believable agent-based simulation systems share the goal of modeling human behavior even if it does not represent optimal decision-making (Bates 1992; Marsella and Gratch 2009). However, many of them are primarily concerned with believability to humans through interaction (Tambe et al. 1995). Here, we are concerned with behavior that is sufficiently similar to humans to produce valid simulation results.

Individual Agents and Teams

Our testing makes use of autonomous agents that collaborate on team tasks to meet their goals. Each individual agent is unique, with parameters such as tireability drawn from distributions of random variables. This increases diversity in the agent population to avoid unrealistic group behaviors that can occur when all the agents act in the same way.

The agents are implemented following a standard model of beliefs, desires and intentions (BDI) in the SPARK agent framework. Each agent maintains a set of intentions, or procedure instances it is currently executing. At each time cycle, it picks a current intention, and either decomposes it into subtasks or performs a primitive action in its environment. It then gathers information about the environment to maintain its intentions in the next cycle. This approach allows the agents to balance time spent deliberating about plans and executing them.

Each agent interacts with its environment through a VNC-based controller for virtual machines developed by Skaion Corporation, and logs in to a virtual machine that controls a host machine on the system under test. This virtual machine provides the execution and sensing environment for the agent. Our agents operate real software on a windows XP desktop, generating realistic network traffic as they go about their tasks and other activities.

We created several specialized agent types that handle different tasks in the scenario as we describe below, but first we describe ranges of beliefs and attributes that are common to all agent types.

Physiology

Human security actions are strongly influenced by their emotional and physical state, including hunger and fatigue. In our current system we model some of the wide body of research on human performance under varying levels of fatigue, which is a major factor both in the propensity to make mistakes and to recognize unexpected behavior in the environment. We are developing models of emotion and bounded rationality to be incorporated in future versions of our agents. We base our models of fatigue on research on human performance, that shows an exponential drop-off in performance over time taken on a task. Our agents maintain a generic fatigue variable that increases towards an asymptote over time and is also increased according to tasks performed. This value is then used in computing the likelihood of mistakes in most tasks the agent performs. In our current scenario, agents take a break from work when they exceed a fatigue threshold, switching to leisure activities that decrease the fatigue parameter, and that may be away from the desktop or may involve web browsing.

Team structure

Many of the attributes and beliefs shared by all agent types concern their position within the organization, team and social network. This includes the agent's supervisor and other agents from which it may request different kinds of help or information.

Scenario

Before we describe the individual agent types we created, we briefly describe the scenario that motivated them and

that we use throughout this paper. A logistics support team is collating information from various web sources and storing it in a cloud-based spreadsheet. Each worker is responsible for a set of spreadsheet cells and communicates with the team manager about completed tasks. Workers periodically review values they entered and call IT support to check for possible faults if they notice any discrepancies. When they take breaks, the agents may visit other web sites. While visiting a site for work or leisure, one user succumbs to a phishing attack that corrupts data on the team spreadsheet. As other workers notice values being changed, they call IT and suspend their work until IT reports the problem is resolved, when they begin by re-entering the correct data.

Individual agent types

Within any organization there is a natural division of labor and specialization of agents into distinct roles with distinct competencies. We divide agents into distinct types, characterized by the set of actions they know how to perform and ranges of possible competencies over these actions, which control their likelihood of making mistakes, along with fatigue and stress level. Agents of different types are likely to share many of the capabilities and all our agents inherit from a base class. The Scenario Director is present in any scenario. The main agent types in the current scenario are the Manager Agent, Worker Agent and IT Agent. In some versions we include cloud administration agents and attacking agents, but they are not used in this paper. Figure 1 shows a typical small organization including a Manager, an IT Agent and several Worker Agents, with links showing recent messages between agents and indicators for each agent's current task load and state (working normally, surprised or resting).

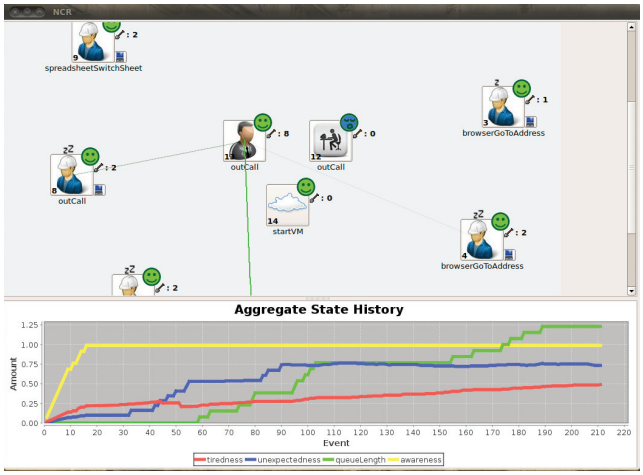

Figure 1: The visualizer displaying a simulation with six workers, one manager, one IT support agent and one cloud support agent. Links show recent communications.

The **Scenario Director** ensures that any pre-determined events in the scenario unfold as expected. It is an agent in order to share the agent framework for communication, but does not model other agent characteristics such as fatigue. It monitors the scenario and ensures events occur as desired either by enacting them directly or sending messages to other agents. In our scenario, the director provides Manager Agents with a range of tasks to perform.

The **Manager Agent** maintains a set of tasks to be performed by a team of workers, dividing them among workers in its team, monitoring progress and re-assigning tasks as appropriate. The main task of the **Worker Agent** is to process information from various sources and enter it in a cloud-based spreadsheet, as assigned by the Manager. The Worker visits a different web site for the information needed to fill each cell and can ask other Worker Agents to perform the task if its tiredness level exceeds a threshold. Workers periodically check that spreadsheet values remain as they assigned them and will call the IT Agent if the value has changed. However, with a small probability the worker makes an error when setting the value. The agent receives a set of tasks from a Manager Agent, sending a message to the Manager as each task is completed.

The **IT Agent** has specialized skills for trouble-shooting desktops. It responds to calls for assistance from worker agents and then connects to their virtual machine to investigate. In our current experiments, computer problems are simulated and the IT Agents do not have real trouble-shooting capabilities. However, when a simulated problem develops, and IT Agent is required to log on to the desktop and declare the problem fixed before the worker can resume. Each agent has a parameter for technical competency, that determines the probability with which an IT Agent will successfully fix a problem. This treatment is sufficient to explore the effect of the IT Agent on the overall behavior of a system under attack, as we show empirically below.

Cyber testing infrastructure

The agent system is designed to support hundreds of agents, each controlling a separate virtual desktop, although many agents may themselves run on the same host. The infrastructure handles both process management (start/stopping processes) and communication See Figure 2.

Process Control/Management

Process management is handled by the process Controller. The Controller receives messages to start or stop a given process (Middleware, Agent, Logger, Visualizer or Scenario Director) on a host machine. The Controller runs as a server/service on each host machine that runs any control process. On starting up, each control process sends a registration message to the host Controller process. The order in which the processes are started is determined by a process called Commander. The Commander sends start/stop control messages to the Controller in a particular order, waiting for one to finish before starting another, in order to respect the dependencies between the processes and ensure all are running before starting the scenario. Similarly, the Scenario Director sends message to the Controller to initialize all the agents. Once the agents are initialized, the Scenario Director begins the scenario. Control and correct ordering of both initiation and termination is vital in a large distributed system such as ours.

Middleware and Subscribers

We model agent to agent communication by email or phone, although the infrastructure allows for any number of communication methods. All agent to agent communication is routed though a process called the Middleware.

The Middleware includes the Router, Directory Service and Subscription Service. The Router looks up the recipient Agent in the Directory Service, sends the message to the Agent, sends a clock tick (preserves order among dependent messages), and then adds the message to the Subscription Service dispatch queue. The Directory Service maintains a list of all the agents on the host machine and any agent it obtained from another Middleware's Directory Service. When an agent starts up it will register itself with the Directory Service on the host machine. Lastly, the Subscription Service is used by processes that monitor all communication and log traffic from all agents on the host machine, such as the Logger process. A Subscriber process registers with the Subscription Service, similarly to the Agent Directory Service registration.

There are currently three Subscriber processes; The Logger, Visualizer, and Scenario Director (also an agent). The Logger saves all messages received from the Middleware in a data base for later analysis. The Visualizer uses a subset of all the messages to create a live visualization of the scenario, or can replay a session from the data base.

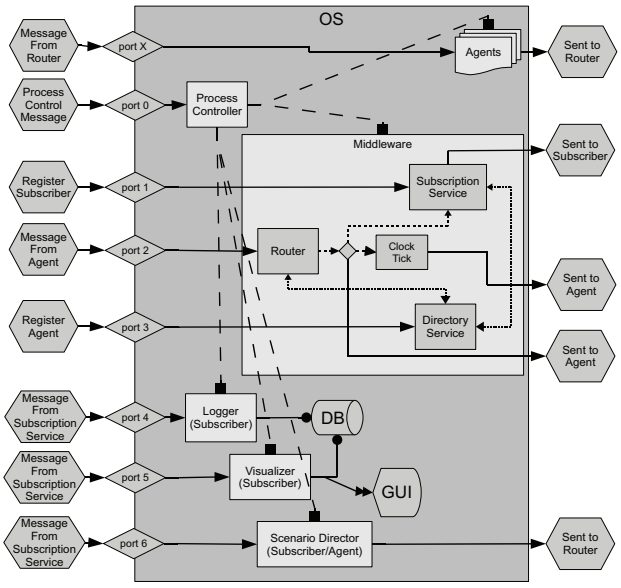

Figure 2: Infrastructure

Agent API and Virtual Desktop

Our current agents are implemented in SPARK (Morley and Myers 2004), although a new framework is under development that combines an associative memory with logic-based action reasoning. An API for the agents to communicate and broadcast state information is written in Java. Since SPARK agents are written in Python, they are executed as Java programs via Jython. When messages are sent between agents, the message body is a SPARK expression since the Agents do not support parsing text or other forms. Messages are sent by the API to the host Middleware.

The agents can either simulate a desktop environment or use an API to Skaion's VNC-based controller for virtual machines. The Skaion controller provides application-level functions, such as opening a browser window and checking for email messages, implemented through mouse and keyboard control and with simple image recognition. It is important that the activities of the agents create realistic network traffic, including communication delays, and operate software in the same way that human agents would. In cases where operating such software is beyond the cognitive capabilities of the agents, we make use of separate semantic messages in an auxiliary network. For example, when agents use email to pass information about task completion, a real email message is sent through the network under test and at the same time, a message is sent between the agents in the auxiliary network containing the semantic content of the message. The recipient may not read the semantic content until the email arrives, allowing us to model the timing and network impact of the messages without parsing their contents.

Empirical Results

Our agent system is currently undergoing testing on a cyber testing range using the cloud spreadsheet scenario. We present here two initial experiments to validate the agents and explore the potential benefits of incorporating human simulations. The first considers the resilience of the overall system to attacks in terms of accomplishing the group mission. The second explores emergent properties of the overall network based on simple actions of the agents.

Resilience to attacks

Although human behavior is arguably the most common source of network vulnerability, humans are also able to respond to unexpected changes in their environment by finding new ways to achieve their goals. Our agents mimic this capability in part with a reactive planning component that can respond to unexpected situations by re-planning to achieve goals in a different way. The alternative behavior must still be expressible in the agent's library of actions, but it may escape the brittleness of rigidly scripted behavior.

The goal of the reactive planning engine also provides an opportunity to measure the extent of damage caused by an attack in terms of partial achievement of the team goals. This is a more salient measure than the penetration of the attack or loss of resources. The first experiment demonstrates this in a simple way by introducing alternative web sites that can provide the information needed to fill in a cell and independently modeling web sites that are knocked out through an attack. When a Worker Agent encounters a site that is unavailable, its plan to fill in the spreadsheet fails. However, it is able to find another plan using one of the alternative sites. There is a time penalty to accessing the first site and switching plans, but this part of the overall goal can still succeed.

Figure 3 shows the proportion of spreadsheet cells that are filled after a fixed time limit when a certain percentage of the web sites have been disabled, chosen randomly. The three curves show the proportion filled when there are one, two or three sites that contain the information needed for each cell respectively. When 50% of the sites are removed, 6 of 10 goals are achieved on average when there is only one site available per cell, but 9.5 goals when there are three sites available. The more sites available, the greater the likelihood that the agent can find an successful alternative to a plan that fails.

A set of scripted agents might have the same behavior as the agents with only one potential site in all cases. If we were not able to simulate and measure the overall level of goal achievement, the measure of the attack would simply be the number of sites made unavailable, failing to benefit from the agent's abilities to re-plan.

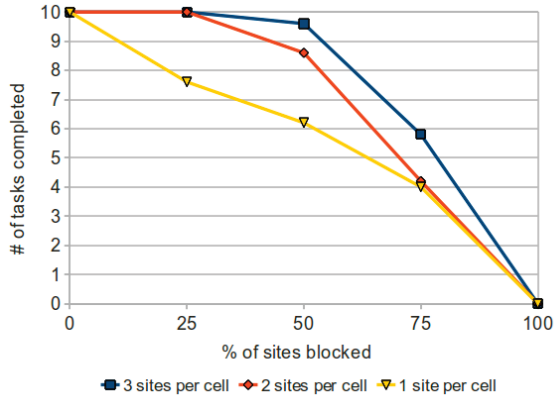

Figure 3: The proportion of tasks completed by a cut-off time as a function of the number of web sites that are removed by an attack.

Emergent network behavior

In many cases, complex behavior in network traffic results from relatively simple behaviors of the individual agents. This is one reason that agent-based simulations to test security have long-term promise compared with statistical traffic generators. Our second experiment demonstrates this effect in the scenario we have followed in this paper. In the scenario, all the Worker Agents call one IT Agent when they notice a problem with the spreadsheet, overloading the IT Agent and causing a build-up of requests under a range of conditions for the IT Agent's speed and rate of tiring. First we show that the IT Agent's rate of tiring has a disproportionate effect on the time that the team takes to complete the spreadsheet because of the bottle-neck position the agent holds in the organizational structure. Second, we show that this leads naturally to an overall network that switches between inaction and bursts of action, as the IT Agent clears the way for a critical mass of Worker Agents in between being repeatedly overwhelmed. This apparently chaotic nature of the network traffic is again a consequence of the organization structure for the task.

Figure 4 shows the average cumulative traffic generated over time for three different configurations of the IT Agent. In the upper line, the IT Agent never gets tired. In the second, it gets tired every forty time steps and takes a break for X time steps. In the third, it becomes tired every ten steps and takes a break for Y steps. We use the cumulative traffic as a measure for the amount of work done. When the agent is tireless, this rises steadily until leveling off as all the Worker agents achieve their tasks. In the other cases it also rises quickly at first until reaching a point where the IT agent rests while a backlog of requests builds up. When the agent rests every forty steps, it eventually clears the backlog of problems caused by the insider attack and the team reaches the maximum cumulative level. When the agent tires every ten time units, it never clears the backlog and the team's progress is very slow. Notice that the graph shows the cumulative work done by the whole team of seven agents, but it is strongly affected by the tiredness on one agent because the organizational structure makes the agent critical when there is an attack.

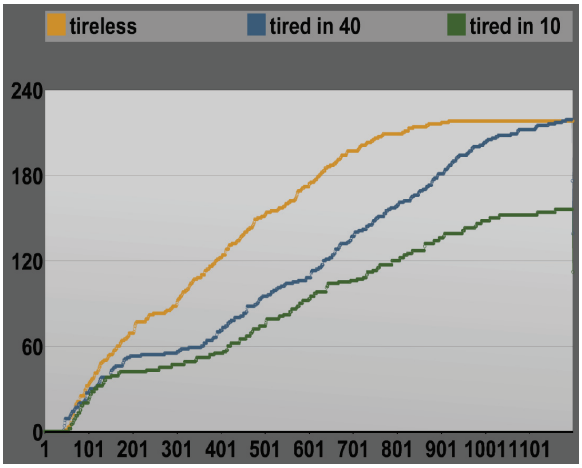

Figure 4: Average cumulative traffic generation for an agent team based on different values for the rate of tiring and recovery rate for the IT agent.

Figure 5 shows the traffic generated over time in a trial where the agent tires every forty units. A number of distinct stages can be seen. Initially there is no traffic as Worker agents log on and look at their tasks. Then traffic rises steeply as they begin to fill in cells, only to drop off as they encounter problems and wait for the IT Agent to check their desktops. This is followed by short bursts of activity that eventually become more pronounced as the IT Agent repeatedly clears the backlog to allow Workers to continue and then is overwhelmed by new problems. This kind of behavior, that switches between discrete modes at distinct times, is difficult for statistical traffic generators to duplicate. In our approach it arises naturally from agents with relatively simple rules and their interactions with each other and their environment.

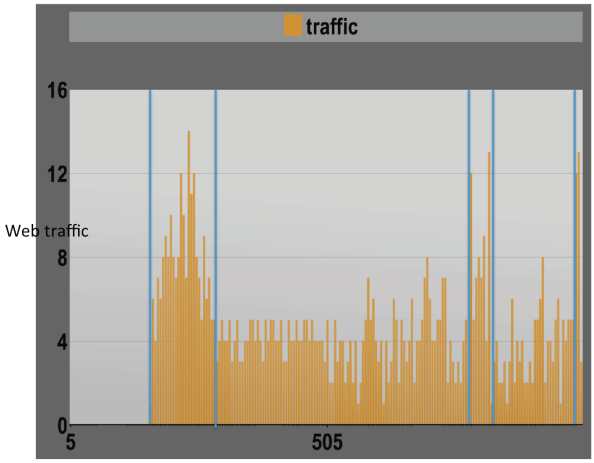

Figure 5: The bursty nature of the traffic generated by the relatively simple agent behaviors of the scenario.

Discussion and Future Work

We presented an agent system designed to test cyber security systems by modeling human behaviors. In particular our agents make mistakes, include simple models of human physiology, work in teams and use reactive planning to respond to their environment and recover from unexpected events. They generate network traffic and create vulnerabilities as they operate standard windows software. We described a typical scenario that models an information gathering task. Since the agents adopt a team goal we are able to use that as a metric of the success of an attack rather than potentially irrelevant measures of penetration and disabled resources. We showed empirically that the agents are more stable to attacks if they can find alternative plans when resources are disabled. We also showed that the team organization can be very important and demonstrated complex traffic patterns that arose from relatively simple agent behavior.

We are improving our agent testing platform in several ways. We are incorporating computational models of emotion based on appraisal and associative networks (Ortony, Clore, and Collins 1988; Bower 1981) that will allow us to better model behavior that has an emotional component, such as performance under stress or fear. Although several computational models of emotion exist (Marsella and Gratch 2009), they do not integrate both associative memory and appraisal with cognitive faculties such as planning. We also plan to incorporate models of bounded rationality, such as the framing effect and patterns of risk seeking and aversion (Kahneman and Tversky 1979; Tversky and Kahneman 1981), that may explain observed behavior as people use security tools. In addition to security testing, we are also exploring the use of these agents in multi-agent behavioral simulations, e.g. (Zyda, Spraragen, and Ranganathan 2009), where planning capabilities combined with physiological models can lead to more faithful simulations.

To help experimenters rapidly instantiate new experiments we are also creating libraries of typical users whose response profiles and abilities can be customized. Based on demonstrations of the importance of the organizational structure, we also plan to create libraries of typical organizations that are operating in a secure environment. This platform is currently being deployed on a cyber testing range as a tool for experimenters to create more realistic tests. We expect to incorporate the lessons from this deployment in future development of the platform.

References

Bates, J. 1992. The nature of characters in interactive worlds and the oz project. Technical report.

Blythe, J.; Camp, L. J.; and Garg, V. 2011. Targeted risk communication for computer security. In *Intelligent User Interfaces*.

Bower, G. 1981. Mood and Memory. *American psychologist*.

Bratman, M. 1987. Intention, plans, and practical reason.

Cranor, L. F. 2008. A framework for reasoning about the human in the loop. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, 1:1–1:15. Berkeley, CA, USA: USENIX Association.

Crawford, M. 2006. Whoops, Human Error Does it Again.

Dourish, P., and Redmiles, D. 2002. An Approach to Usable Security Based on Event Monitoring and Visualization. In *New Security Paradigms Workshop*.

Kahneman, D., and Tversky, A. 1979. Prospect theory: An analysis of decision under risk. *Econometrica* (47):263–291.

Marsella, S., and Gratch, J. 2009. EMA: A process model of appraisal dynamics. *Cognitive Systems Research*.

Mitnick, K. D. 2002. *The art of deception: Controlling the human element of security*. Wiley.

Morley, D., and Myers, K. 2004. The SPARK Agent Framework. In *AAMAS*.

Ortony, A.; Clore, G.; and Collins, A. 1988. *The Cognitive Structure of Emotions*. Cambridge: Cambridge University Press.

Tambe, M.; Johnson, W.; Jones, R.; Koss, F.; and Laird, J. 1995. Intelligent agents for interactive simulation environments. *AI magazine*.

Tversky, A., and Kahneman, D. 1981. The framing of decisions and the psychology of choice. *Science* (211):453–458.

Whitten, A., and Tygar, J. D. 1999. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*, 14–14. Berkeley, CA, USA: USENIX Association.

Wogalter, M. 2005. Communication-human information processing (C-HIP) model. *Handbook of human factors in Web design*.

Wright, C.; Monrose, F.; and Masson, G. 2004. HMM Profiles for Network Traffic Classification (Extended Abstract). In *Proceedings of the Workshop on Visualization and Data Mining for Computer Security*.

Zyda, M.; Spraragen, M.; and Ranganathan, B. 2009. Testing behavioral models with an online game. *IEEE Computer* 42(4):103–105.