Network Security Testing Using Mobile Agents

Author: T. Karygiannis

Affiliation: National Institute of Standards and Technology

Postal Address: NIST
Information Technology Laboratory
Building 820, Room 426
Gaithersburg, MD 20899

Telephone: 301-975-4728

Fax: 301-948-0279

Email: karygiannis@nist.gov

Abstract:

This paper describes a prototype security testing tool that is currently under development at NIST. This prototype tool uses autonomous mobile agents to perform distributed, fault-tolerant, and adaptive network security testing. The security testing prototype is compared to existing methods, the design goals are outlined, its testing methodology is described, and the advantages and disadvantages of using mobile agents for security testing are discussed.

Abstract

This paper describes a prototype security testing tool that is currently under development at NIST. This prototype tool uses autonomous mobile agents to perform distributed, fault-tolerant, and adaptive network security testing. The security testing prototype is compared to existing methods, the design goals are outlined, its testing methodology is described, and the advantages and disadvantages of using mobile agents for security testing are discussed.

1 Introduction

Our reliance on computers to operate safely and reliably in environments ranging from air traffic control and telemedicine, to electronic commerce and telelearning, underscores the need to design, develop, and maintain secure computer systems. As new products are continuously being introduced into the marketplace, a significant number of security vulnerabilities and vulnerability classes are regularly being detected and exposed during the operational lifetime of a product. This trend is likely to continue given the time-to-market pressure, complexity of the security evaluation process, the unlimited number of system configurations, product revisions and upgrades, and varying levels of system administrator expertise. Although there are a number of security tools that are designed to detect system vulnerabilities, these tools have a number of limitations, such as:

- Network security testing and monitoring is typically assigned to a single central security monitor. The failure of this central security monitor will render the system unable to perform security testing and vulnerable to attack[1,2].
- Host-based monitors and integrity checkers can be disabled and audit trails corrupted by intruders.
- As the size and complexity of the network increases, so does the computational burden and data logging requirements placed on this central security monitor.
- As new vulnerabilities are discovered and security advisories are issued, the test suites of these security tools quickly become outdated and new tests are often not available until the next software release.

This paper outlines a test methodology and prototype security tool that attempts to overcome these limitations by using autonomous mobile agents.

2 Design Goals

The objective of this project is to develop a network security testing methodology and a prototype tool that will meet the following design goals:

- The security testing tool should be easy to maintain, upgrade, and configure.
- The security testing tool must be able to test and monitor heterogeneous networks.
- The test methodology should be fault-tolerant.
- The security testing tool should scale well as the number of computers in the network increases.
- The security testing tool should be adaptive.

3 Network Security Testing Tools

Security testing and monitoring tools are typically implemented as either central security monitors or as host-based integrity checkers. The limitations of central security monitors and host-based integrity checkers are summarized in sections 3.1 and 3.2.

3.1 Central Security Monitor

A central security monitor is typically assigned the responsibility of performing the security testing and monitoring of a network. A failure of this central security monitor, however, renders the system unable to perform security testing and protect the network from attack. This failure could be caused by a hardware malfunction, the loss of a network connection, an attack from a malicious cracker, or be the result of an organized "infowar" attack. For example, Figure 1(a) below shows a fault-free central security monitor that has detected a security problem in node N_2. The shaded nodes in Figure 1 represent a suspicious, compromised, or faulty node. If the central security monitor fails, or is compromised, as shown in Figure 1(b), then the security monitor would not be able, or could not be trusted, to detect security problems in the rest of the network. A faulty or compromised security monitor may no longer be able to monitor the rest of the network, may be providing erroneous data due to some hardware failure, or may even try to actively deceive the network by lying or allowing intruders into the system. Clearly, this "command center" approach to security testing and monitoring is not very fault-tolerant. In addition to not being fault-tolerant, centralized security monitor implementations do not scale well. As the number of computers and other resources on the network increases, so does the computational load on the central security monitor. As its performance decreases, the security testing intervals would increase and, as a result, widen the window of opportunity for an intruder to penetrate the system.

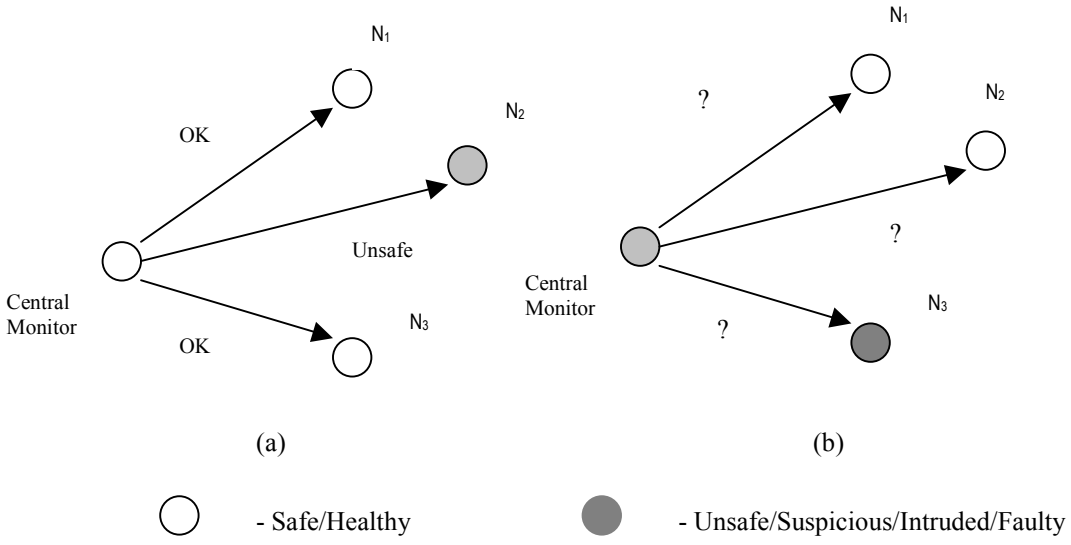

Figure 1. A centralized network security monitor.

3.2 Host-based Integrity Checkers

Host-based integrity checkers reside on the node being monitored and report changes to critical system files or suspicious activity back to the node's system administrator, a central monitor, or broadcast the test results to the rest of the network. Host-based monitors can run as background daemons or as scheduled processes. An unauthorized user that gains control of a computer can act maliciously and convince other parts of the system that it is safe when, in fact, its security has been compromised. An intruder can disable security monitoring daemons, deactivate "cron" jobs, disable the reporting utilities, and erase audit trails. Thus, a computer cannot always be trusted to diagnose itself with a host-based integrity checker alone. In Figure 2(a), for example, a safe node N_1 runs a host-based security test and reports the results to the network or to a central security monitor. A compromised node N_1, as shown in Figure 2(b), however, could deceive the rest of the system by disseminating erroneous information. Alternatively, if a node running a host-based integrity checker reports its test results to a central processor, and the central processor has failed, then the rest of the nodes in the network would never be informed of a potential security problem. The central security monitor implementation is not fail-safe. The host-based implementation is fail-safe for hardware failures, but not for security breaches as the intruder can disable the host-based security monitors.

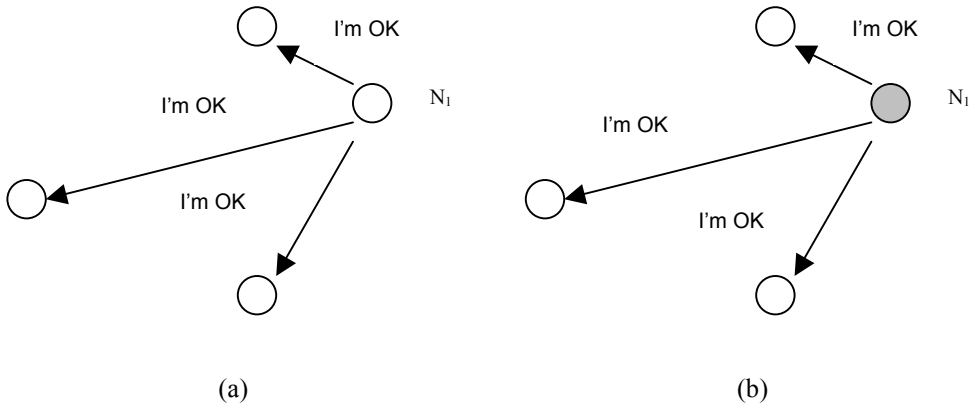

Figure 2. A compromised system running host-based security tools.

4 Network Security Testing using Mobile Agents

In order to overcome many of the limitations of central security and host-based monitors, a prototype tool is being designed and developed to demonstrate the benefits of using mobile agents to perform security testing. This section describes the prototype design and provides the rationale and benefits of using mobile agents to perform security testing. The mobile agent security tool prototype is currently being implemented using IBM's Aglet Workbench and JDK1.1.4 and runs on Windows95 and SPARC/Solaris2.5 machines[3].

4.1 Fault-Tolerance and Scaleability

Mobile agents' ability to autonomously travel through the network adds a new level of fault-tolerance to security testing tools. Their ability to travel through the network and carry data along with them enables the agents to hide data, code, and security-relevant information from potential intruders. Moreover, the security testing of the network is not vulnerable to a single point of failure and the security testing can continue even if individual nodes fail or become unavailable[2,4]. For example, an agent can compute and encrypt the signature of critical files on one host and travel to another host or set of hosts and report the results.[1] The agent itself can also be encrypted. Since the agent can carry both data and code with it, the security tools are more difficult to disable, or circumvent. The problem of preventing snooping or tampering with an agent's internal state is still an open research problem[5].

The addition of new computers to a network places an additional computational load on central security monitor implementations. A distributed security tool that uses mobile agents, however, can overcome this limitation by dispatching or cloning new agents as computers are added to the network, and thus share the additional workload and offer better scaleability. The ability to have agents travel through the system and execute their code using the hosts resources also allows for dynamic load sharing.

4.2 Adaptive Testing

Agents can travel to nodes based on the time of their last visit or a predefined schedule, based on the network load, a host's computational load, or based on messages received from another agent. Agents can also decide on their own what path to take and what actions to perform as they gather data from the nodes they visit. If an intruder blocks the hosting capability of a node, then the agent can go to another host and report the problem. Cooperating agents can help reconfigure the network to deny network services to certain nodes until they have been confirmed to be in a safe state. Agents can monitor network events and cooperate with host-based agents. For example, if an agent detects suspicious activity on one computer and notifies the rest of the network, the other computers may decide to challenge the suspicious node by not giving it the rights to mount some files it previously had rights to, or by denying it certain network services, or by reconfiguring the network until the suspicious node returns to a safe state.

4.3 Agent Control and Cooperation

Meta-agents can monitor, coordinate, retract, and dispatch agents as needed. An agent can be dispatched to a specific host on the network or can be given an itinerary of hosts to visit and test. Once an agent completes its tests, it can send messages to call other agents to help with the testing by performing more detailed tests, if necessary, or to perform other tests. Agents can transmit beacons to meta-agents to verify that they reached their destinations and have completed their tasks. Agents can broadcast "alarms" to notify other agents of potential security problems. All communications are handled transparently by proxies. Agents can have multiple proxies since the failure of the proxy's host would cut off communication from an agent that is traveling through the network. The ability of agents to travel, cooperate, and communicate make it much more difficult for an intruder to disable or circumvent security testing and monitors. This inter-agent control and messaging, however, also introduces a communication overhead that is not found in central security monitor or host-based integrity checker implementations.

[1] The use of cryptography with the agents has not yet been implemented in the prototype.

4.4 Maintenance

Agents are constructed as lightweight processes, so that each process tests a single vulnerability. As new vulnerabilities are detected and tests for these vulnerabilities are developed, new agents can be added to the test suite. As the system configuration changes, some agents can be retracted or disposed if they are no longer needed. Test suites can be fine-tuned for each individual node depending on its configuration. This increases the efficiency of the testing as tests are performed only where and when they are needed. A lightweight agent architecture makes the test suite very configurable, especially for a heterogeneous environment.

4.5 Mobile Agent Security Issues

The use of mobile agents for network security testing and monitoring introduces new security concerns and vulnerabilities. Many agent qualities make them an ideal attack mechanism. Hosts must be protected from malicious agents, agents must be protected from malicious hosts, and agents must also be protected from other malicious agents. Agent attacks can include spamming, spoofing, and denial of service attacks. Spamming can occur by flooding a service with requests, denial of service attacks can be mounted by excessive cloning of agents for the purpose of draining computational resources from a host, and spoofing can occur when an agent or agent system falsifies its identity to get access to information or services[5]. The encryption of agents and their messages, and the use of digital signatures provide additional security against these threats, but more research in the area of agent security is required before agents can be safely deployed in critical applications.

4.6 Advantages

There are several advantages to using mobile agents to perform distributed security testing. Some of these advantages are:

- Mobile agents add fault-tolerance. The network is not vulnerable to a single-point of failure.
- It is more difficult for an intruder to disable security tests and monitors when they are distributed throughout the network.
- Mobile agents can take advantage of the inherent parallelism of large networks to offer performance improvements over conventional centralized security monitoring by distributing the security testing workload over the network.
- Distributed security testing scales well as more agents can be added when new computers are added to the network. Maintaining and upgrading the security test suite can be accomplished by dispatching or cloning new agents and retracting or disposing of old agents.
- Agents can be adaptively assigned to different tasks spanning a range of responsibilities from intrusion detection and diagnosis, to reconfiguration and recovery.
- Agents can be developed to run on heterogeneous computer systems and take advantage of the built-in agent communication capabilities.

4.7 Disadvantages

Some of the disadvantages of using mobile agents are:

- Mobile agent tools are still new and may have security bugs and vulnerabilities that are yet unknown.

- Network test suites tend to be relatively large. Managing many lightweight agents introduces additional communication and control overhead.

- Mobile Agents are not a mature technology and most agent development tools are alpha or beta versions.

- Although an agent's ability to travel throughout the network introduces fault-tolerant properties to the security tool, it also exposes the agents to new security threats and risks that host-based security tools do not encounter.

- The agent host environment must be installed and maintained on each node.

5 Conclusions

The prototype security tool is being designed to demonstrate the fault-tolerant and adaptive capabilities of mobile agents. The distributed nature of the mobile agents makes it more difficult to circumvent or disable security testing than in a central monitor or host-based security testing approach. The lightweight design of the agents themselves allows agents to be easily updated or retracted as new vulnerabilities are discovered or old vulnerabilities are patched. Mobile agents seem to be well-suited for network management applications in heterogeneous environments. An agent context "browser plug-in" would help network administrators deploy agent systems more easily. The encryption of agents and their messages, and the use of digital signatures provide some measures of security, but additional research is needed in the area of the security of the agents, the agent contexts, and the agent protocols themselves.

6 References

[1] Jai Sundar Balasubramaniyan, Ganesh Krishnan, Eugene Spafford, Karyl Stein, Aurobindo Sundaram, *Software Agents for Intrusion Detection*, COAST Laboratory Technical Report, Department of Computer Sciences, Purdue University, May 15, 1997.

[2] Mark Crosbie and Eugene Spafford, , *Active Defense of a Computer System using Autonomous Agents*, COAST Laboratory Technical Report, Department of Computer Sciences, Purdue University, February 15, 1995.

[3] Danny B. Lange and Mitsuru Oshima, *Java Agent API: Programming and Deploying Agents with Java*, to be published by Addison-Wesley, Fall 1997, a working draft, "*Programming Mobile Agents in Java*," is available at http://www.trl.ibm.co.jp/aglets/whitepaper.html

[4] Michael Schroeder and Gerd Wagner, *Distributed Diagnosis by Vivid Agents*, Inst. F. Rechnergest. Wissensverabeitung, Univ. Hannover, Inst. F. Informatik, Univ, Leipzig, Technical Report, 1997.

[5] Gunter Karjoth, Danny B. Lange and Mitsuru Oshima, *A Security Model for Aglets*, IEEE Internet Computing, 1997.