

Phobos: An Agent-Based User- Authentication System

Robert Ghana-Hercock, *BTextact Technologies*

We all trust other people to some degree—we trust the driver behind us to brake in time and our 747 pilot to know his or her job. Yet when it comes to the Internet, we enter a fog of paranoia and trepidation. The computing industry and network suppliers are busy hardening their services against hackers and script kiddies, yet

do we understand where the real threats are? The point is, how many system administrators and public-key infrastructure experts can companies afford to employ or even recruit? Considering that devices and network channels will soon number in the billions, we would be wise to doubt enough skilled labor will exist. More important, even current threats are overwhelming IT companies because they have insufficient human resources to install patches to known security bugs.

Securing IT services is a complex, continuously evolving battle between defensive and offensive strategies. (See the sidebar “Securing the Network” for more details.) This process now focuses on authorizing legitimate users of IT resources because the human element is the weakest link in any security architecture. In addition, recent reports have clearly indicated internal staff’s central role in undermining system security through errors or intentional activity. Such internal attacks are difficult to defend against with firewalls and password measures.¹ The industry has good encryption techniques and strong public-key mechanisms, but if we cannot authenticate a system’s end users, the whole security strategy fails.

To solve this problem, the BTextact Technologies Intelligent Systems Laboratory has developed the Phobos agent architecture. Phobos uses a distributed team of cooperative autonomous agents to collectively authenticate user access requests. The advantages are that the agents can query multiple information sources to select the level of trust to delegate to a user and that n agents must concur to authenticate the user, hence increasing overall security. Pho-

bos provides numerous security services to automate user authentication and trust-management processes.

The Phobos system

Our premise is that software agents are ideal for managing user and system authentication. Related work on security-authentication agent control has strongly demonstrated this design approach’s validity for user authentication.^{2,3} Researchers have frequently proposed cooperative software agents as a tool for information-processing tasks such as e-commerce transactions⁴ and workflow modeling and as personal assistants.⁵ However, relatively little research has considered their role as an element of network security.^{6,7} (For more on approaches to user authentication, see the “Related Work” sidebar.) Indeed, the industry sees some agent classes, such as mobile agents, as a serious security threat to any host network.⁸

Our agent approach’s central concept is that by linking the sensory and intelligence capabilities of numerous agents distributed across a network, we can amplify the network’s ability to determine the trustworthiness of users and increase the system’s resistance to attacks. Specifically, through social cooperation, agents can benefit from the group’s combined defensive capabilities and knowledge base. In addition, a central requirement for Phobos was to support security services within peer-to-peer (P2P) and ad hoc networks, which require a distributed design approach. Some specific properties of a software agent system that match the requirements of a distributed security authentication platform

Managing the necessary public and private keys in a large organization is a serious challenge.

Software agents can be an adaptive and responsive mechanism for managing users trying to connect to network resources.

Securing the Network

Three major requirements exist when securing a network: intrusion and virus detection, firewall management and packet monitoring, and management of authorized user access. Many commercial software and hardware systems cover the first two domains. These systems have varying degrees of success, however, and too often excessive reliance results from assuming that a perfect firewall or antiviral system is in place.

User authentication, on the other hand, has been neglected in terms of the technology level commonly applied. The major change has been from a simple password login to biometric methods based on fingerprint or iris-scan identification. These technologies are maturing, but deployment costs are still prohibitive. Alternatively, we can issue users smart cards with hardware encryption. This is a secure solution widely used in military and sensitive commercial areas, but it is expensive and costly to manage. Also, a lost or stolen card might result in a serious security breach. Even with biometric and smart-card solutions, managing users' personal data and requests for assistance still requires manual intervention. Issuing digital certificates is easy, but managing their life cycle—that is, revoking or reissuing them and associating them to the proper user or service—is vastly more difficult.

Current solutions to network user authentication are frequently inflexible, centralized, and expensive to implement

and maintain, and they provide inadequate defense.¹ IBM's recent Autonomic Computing initiative (www.research.ibm.com/autonomic) provides a clear example of the realization that future progress in computing services requires vastly greater degrees of automation. However, few products offer any real degree of automation for integrated security services. Companies also face increased threats from internal and external attacks and the escalating costs of managing the increasingly complex security infrastructure required to defend themselves. (The market for security services is growing at approximately 2 percent per year.) One US company currently receives 14,000 calls a month from employees who have forgotten their passwords and require a manual reset.² Each call costs anywhere from \$25 to \$200 for an employee using multiple systems or software programs.

References

1. L. O'Gorman, *Human Authentication—The Achilles Heel of Secure Systems*, Avaya Inc., 2002, www.research.avayalabs.com/user/logorman/UA_Stevens.ppt.
2. A. Salkever, "Software That Asks 'Who Goes There?'" *Business Week*, 26 Feb. 2002; <http://online.securityfocus.com/news/339>.

include autonomous goal-driven behavior, reasoning, and the capacity for proactive responses to potential threats.

Developing a distributed user-authentication system requires a process for developing and sustaining trust relationships between users. (Principals might also be other software agents requesting access to a specific service or resource.) In the same manner that a community of individuals creates a network of trust built up over time, so an agent community can grow a network of trust relationships.

Collective authentication

The key to Phobos is collective authentication, where multiple cooperating agents must authenticate a user. To prevent spoofing of the Phobos agents, all agent-to-agent interaction first requires mutual authentication through public-private key exchange. (If the initial request comes from an external, non-Phobos software agent, the system treats it as an untrusted principal).

The agents can combine multiple information sources to develop a user's trust profile and to decide what level of authentication to grant. The trust profile is an XML-format data type that contains two floating-point numeric values (scaled from 0.0 to 1.0). These values represent the agent's inferred degree of trust in

the user and a confidence in the assigned trust level. Each agent's rule-processing module (which I discuss in more detail later) uses this profile to infer the level of authentication to assign a principal.

The agents use a two-tier architecture to distribute authentication messages between agents, with a single master agent responsible for a set of subdomains and a peer-slave agent responsible for actions in each subdomain. The master agent interacts with the network's human administrator and ensures that all peer agents implement the centralized security policy.

The agent system uses existing access control databases within an Intranet environment. So, each peer agent can access and update local user access databases—for example, using the lightweight directory access protocol (LDAP). However, each agent can also maintain its own database of user access details if the application is a wireless or P2P-style network. I am still investigating several outstanding problems regarding how to maintain and merge these databases. In particular, I'm studying how to give agents sufficient autonomy to perform their authentication role while retaining clear administrative policy control.

In the current version of Phobos, each agent maintains a separate user trust profile for the local network domain that the agent

manages. Agents then exchange profiles at an authenticated Phobos agent's request. (Related work within the BT intelligent systems group is developing a system for integrating and securely managing distributed P2P databases of this type.⁹)

An example task

As an example, let's step through a typical user-agent interaction. The user need not be aware of the agent's existence, although future system versions will consider issuing user-agent dialogue as part of the authentication process. In this scenario, a corporate user needs temporary access to a supplier's product database from a mobile device over a wireless-network link. The supplier wants to grant access to authorized customers but with time-limited restrictions and automated checking of their credentials. The user first logs in via a browser interface and submits some preliminary identification—such as user ID and customer number. The request passes via an SSL (secure sockets layer) HTTP link to the supplier's secure Web server, which routes the message to a suitable local Phobos agent. All communications, even local host communications, occur over encrypted SSL channels.

On receiving the user's request, the agent first converts it into an agent communication

Related Work

One industry standard for achieving strong user authentication to network services is the Kerberos system.¹ Kerberos uses a trusted third-party authentication service in which each client trusts the Kerberos central server to authenticate other clients. Time stamps on each client-server communication prevent or at least reduce the chance of a replay attack.

The Kerberos server maintains a database of its clients and their private keys and an encrypted password for users. Any network services or clients requiring authentication must register with the Kerberos server. The private keys are then negotiated at registration. Kerberos can also provide secure session management based on session keys. The Kerberos system has proven to be a useful authentication platform but requires a single centralized server, which is frequently the target of attacks because access gives full network control. It also requires expert administration for its maintenance. Some work is in progress to establish a simpler mechanism for public-key architectures (for example, the Simple Distributed Security Infrastructure²), which would encourage their wider use and ease the management burden.

An important and parallel development is using formal-logic-based knowledge representations to process authentication queries between subjects.³ This work focuses on the inferencing required to assign trust in any open or peer-based large-scale network where the authorization query is expressed in the generic form—Does the set of credentials C prove that the request r complies with the set of local security policies P ?

A further issue regarding authentication between agents is how to measure and evaluate the degree of an agent's trustworthiness. Related work on interagent trust exchange mechanisms indicates a solution based on an advance payment contract between agents.⁴ We could apply this to Phobos by having authenticating agents require a user or an agent to deposit a financial bond before the system grants a service.

Some preliminary agent-based work in this field has already demonstrated the effectiveness of agent-managed security methods, although primarily in intrusion detection.^{5,6} Guy Helmer and his colleagues have demonstrated a network defense system in which software agents monitor low-level network activity and report it to higher-level software agents for analysis.⁷ However, the closest work to Phobos is probably that of Lalana Kagal and her colleagues, which defines a framework for agent-based trust management.⁸ In this system, the agents also manage access control policies and issue authentication certificates to requesting principals, although it appears to have no P2P capability.

In the system Mark Crosbie and Eugene Spafford propose, a similar distributed set of agents monitors network traffic and machine activity, including CPU utilization.⁹ Their system also has agents exchanging anomaly reports, and decisions to raise an intrusion alert are based on the combined evidence from numerous agents. This system uses a form of machine learning

based on genetic programs to recognize new attack patterns. Curtis Carver and his colleagues have used a distributed heterogeneous group of agents as an IDS solution.¹⁰ They focus on a dynamic, adaptive response to varying levels of security threats.

One commercial system that uses a similar Web- and agent-based authentication method is seqID's ProviderTrust Web access system (http://giants.getdigitalid.com/content/pt_rsa_web_access.html). However, this is a single agent system designed to provide authentication within a single intranet domain.

References

1. S.P. Miller et al., "Kerberos Authentication and Authorization System," Project Athena, tech. report, Massachusetts Inst. of Technology, Cambridge, Mass., Dec. 1987.
2. R. Rivest and B. Lampson, *Cryptography and Information Security Group Research Project: A Simple Distributed Security Infrastructure (SDSI)*, Laboratory for Computer Science, Massachusetts Inst. of Technology, 2002, <http://theory.lcs.mit.edu/~cis/sdsi.html>.
3. N. Li, J. Feigenbaum, and B.N. Grosz, "A Logic-Based Knowledge Representation for Authorization with Delegation," *Proc. 12th Int'l IEEE Computer Security Foundations Workshop*, IEEE CS Press, 1999, pp. 162–174.
4. S. Braynov and T. Sandholm, "Contracting with Uncertain Level of Trust," *Proc. 1st ACM Conf. Electronic Commerce*, ACM Press, 1999, pp. 15–21.
5. R. Filman and T. Linden, "Communicating Security Agents," *Proc. 5th Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 96)*, IEEE CS Press, 1996, pp. 86–91.
6. J. Balasubramaniyan et al., *An Architecture for Intrusion Detection Using Autonomous Agents*, Coast tech. report 98-05, Dept. Computer Sciences, Purdue Univ., 1998.
7. G.G. Helmer et al., "Intelligent Agents for Intrusion Detection," *Proc. IEEE Information Technology Conf.*, IEEE Press, 1998, pp. 121–124.
8. L. Kagal, T. Finin, and Y. Peng, "A Framework for Distributed Trust Management," *Proc. IJCAI-01 Workshop Autonomy, Delegation and Control*, 2001.
9. M. Crosbie and E. Spafford, "Active Defense of a Computer System Using Autonomous Agents," tech. report 95-008, Dept. Computer Science, Purdue Univ., Feb. 1995.
10. C.A. Carver et al., "A Methodology for Using Intelligent Agents to Provide Automated Intrusion Response," *Proc. IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, IEEE Press, 2000, pp. 110–116.

language (ACL) message format and fires its reasoning module to process the request. In this case, the system must validate that the user has the necessary clearance to access the requested resource. The agent then passes the

user's details to its own security service module, which returns the user's current status—that is, authorized, unauthorized, untrustworthy, or malicious. If the agent does not know the user, it transmits the request to

another agent in the supplier's network (or n agents via a multicast message).

The receiving agents again attempt to validate the sending agent's authority and identity and then check the user request against

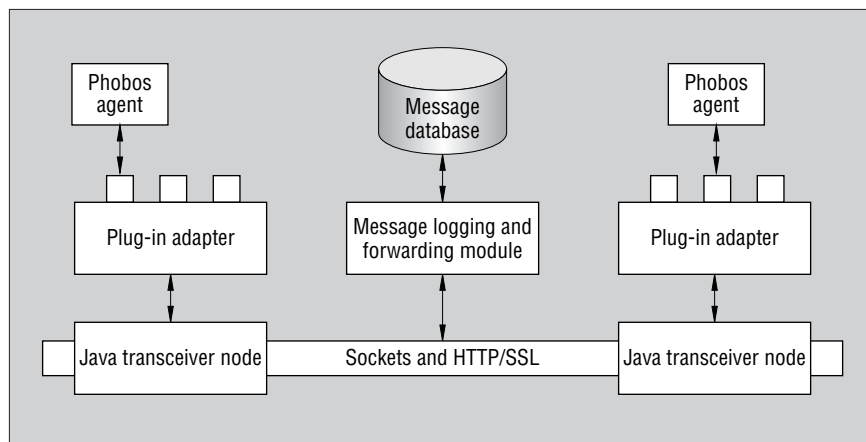


Figure 1. The Phobos messaging layer.

their local database of authorized users. If the current agent grants authorization, it then adds a suitable certificate or key to the original message and transmits the new processed request back to the requesting agent (or to the next agent if this process is performed sequentially).

Once a specified number of agents have authenticated the request, the final processing agent returns the completed message with certificates to the initiating agent. The agent then grants access to the user by returning a new password and link to the requested Web database via the user's mobile device. It also assigns the necessary password and X.509 certificate details to its local database of authorized users with a time stamp on the certificate.

To prevent messages from flooding the network, each message contains the number of checks it requires, thus preventing it from being repeatedly processed. A failure at any point results in either a denial-of-access message to the user or a message sent to the system administrator via the master agent. The master agent also has the authority to initiate contact with an external trusted third-party service to request further credential checks on a user.

By simply increasing the number of agents required to authenticate a user, we can considerably increase the degree of security. A useful analogy is that when a company employs someone, it typically checks several independent references for that employee.

Design assumptions

The Phobos system security therefore depends on several trusted machines. Hence, although any hostile attack might compromise

one or more machines, it should be unable to achieve control over all the machines. (Obviously, a distributed denial-of-service attack can cripple many machines simultaneously, which means we should shield the identity of the agent's host servers.) In particular, because Phobos uses distributed agents, a hostile user will have difficulty accessing the runtime code for all the agents. Because an attacker can communicate only via the established message channels with the agents, the agents can then interpret each message and apply authentication to the requesting agent. In addition, simple updates to the agent's rule set can allow even more complex cross-checking and referring of a user. (Of course, we must tightly administer access to the agent's reasoning system and rules.)

The principal design criteria were to provide a scalable, modular, and highly robust architecture. In particular, if the goal is safe, reliable operation, simplicity must be the design goal.¹⁰ If necessary, we can position numerous agent host machines behind firewalled servers or heavily protected machines.

The system architecture

The basis for Phobos agent components is the JADE (Java Agent Development Framework, <http://jade.cse.it>) agent toolkit, which provides core messaging and agent visualization services. We selected JADE because it offers a lightweight Java agent toolkit with good FIPA ACL (Foundation for Intelligent Physical Agents Agent Communication Language) compliance. Once loaded, the JADE container allows full administrative GUI access to visualize agent messages or introspect the agent's state. The agent on each host must also occupy an acceptable level of local

resources, achieved via a lightweight modular design. To connect services to the network, Phobos uses a middleware layer of Java P2P HTTP servers (see Figure 1). Each agent resides on a single host machine and controls the GUI for the user.

A Phobos system goal was to enable ease of code reuse and robust operation. We achieved this through a highly modular design that cleanly separates the messaging system, Web interface, and agent classes. For example, by using the JADE interagent messaging facility, the agent system can run independently of the peer-based messaging layer, and it is tolerant of intermittent availability of the peer-messaging layer. We are also extending the agent inferencing engine with a modular JavaBean rule-based system to further increase the agents' modularity. The system has a three-layer architecture, starting with the autonomous transceiver nodes that provide the primary messaging channel. Next is the plug-in layer, which lets specific services and agents use the messaging layer. The final layer consists of Phobos agents or Web applications, which provide the system's functionality.

The user interface is an HTML form that lets the user input his or her current security identification data. A standard URL password access panel controls access to this form, thus providing a first line of access control. Phobos returns the response data from the agent to a second HTML page, which provides links to specific resources the user has been granted access to. As part of ongoing development of the Phobos system, the Intelligent Systems Laboratory plans to add more intuitive user interface capabilities in future system versions. In particular, it would be useful if the agent could conduct a text or voice dialogue with the user to obtain secondary authentication information, such as place of birth.

One of the main design specifications for the Phobos system was to enable support for security services in P2P and ad hoc networks. To enable this, the whole system uses a P2P messaging layer based on a network of autonomous Web servers. These servers are lightweight Java HTTP servers that automatically maintain links with any available peer nodes across the network. Through HTTP, firewall tunneling can link nodes at multiple sites or separate intranet domains. Each server/node maintains a dynamic list of available peers and periodically exchanges this list with neighboring nodes.¹¹

This architecture's key advantage is that it is highly robust to individual node failure. So, when users log on to Phobos, they have guaranteed service availability. This P2P architecture is also a key aspect of the agent-to-agent communication system, which also requires highly robust and flexible service. Indeed, many agent systems have relied on centralized message servers that have hindered scalability and acted as a single point of failure.

In addition, the messaging layer contains independent code modules that intercept all message traffic and store it in a secure distributed message database, to provide store-and-forward capability. So, if an agent is offline or busy, Phobos will save the messages and relay them whenever the agent becomes available. As Figure 1 shows, the system isolates the agent from the messaging layer via a plug-in module, which means that we can easily port future versions to a different messaging architecture or service. The plug-in layer also connects to Web-based resources that the agents might need to access, such as LDAP databases of user credentials.

Agent architecture

Figure 2 shows the Phobos agent core architecture components. Next I describe the internal components of a Phobos agent. These include a multithreaded message-handling module, a task coordination engine, a security module, and a local resource management service.

Message Handler

Each agent contains a multithreaded Message Handler module that handles all incoming connections. The messaging service maintains multiple SSL socket connections to the plug-in component, which links the agent to the underlying messaging service. Internal to the message-handling class, separate threads handle each socket, with a buffered message-queuing service. A separate threaded class pops inbound messages from the buffer and passes them to the Task Engine for processing. It also pushes outbound messages onto the buffered queue for processing by the message-handling classes.

Message format

The messages are based on the ACL format (see Figure 3), which defines basic elements (such as sender and receiver agents) and specifies the language and protocols used. Phobos also uses the ACL `contentObject` element to store a serializable object that holds the user's

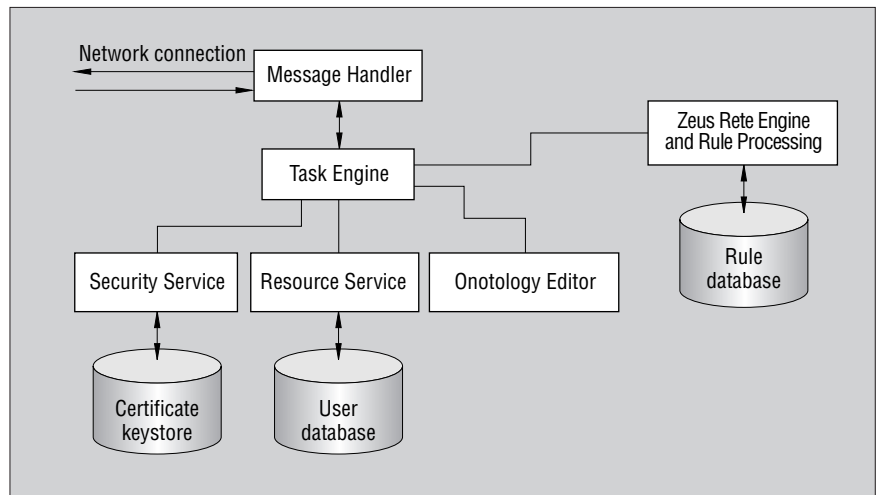


Figure 2. The Phobos agent core architecture. The network communication between the agent and the plug-in class uses a multithreaded SSL (secure sockets layer) socket connection.

```
//ACL Message
(REQUEST
  :sender ( agent-identifier :name
    bob@revolve:1099http://132.000.000.000:8082/agentPlugin/receiveData)
  :receiver (set ( agent-identifier :name
    bob@revolve:1099http://132.000.000.000:8082/agentPlugin/receiveData) )
  :content "User taskObject class : X509 cert :
    r00ABXNyABNkYXRhYmFzZS51c2VyT2JqZWNOB04UlvC4rrsCAAdMAAJpZHQAEkxqYXZhL2xhbmcvU
    ="
  :reply-with broadcast
  :in-reply-to receiver
  :encoding X509
  :language FIPA-SLO
  :ontology agent-security
  :protocol fipa-request
  :conversation-id 1014975952282
```

Figure 3. An Agent Communication Language Phobos agent message including X.509 certificate data. This message format is the base message type transferred between the agents via the peer-to-peer messaging layer in Figure 1.

request data and security details—that is, an X.509 certificate. In addition, if required, the JADE platform provides a local-network inter-agent communication facility. We can visualize these local agent messages via the JADE management interface.

Task Engine

The Zeus agent platform's inferencing core forms the basis of the Task Engine. Zeus (<http://more.btexact.com/projects/agents.htm>) is an open-source multiagent toolkit that provides a library of software components and

tools that facilitate the rapid design, development, and deployment of agent systems. We used seven principal components from the Zeus platform:

The *Coordination Engine* makes decisions concerning the agent's goals—for example, how to pursue the goals and when to abandon them. It also coordinates the agent's interactions with other agents using its known coordination protocols and strategies, such as various auction protocols or contract Internet protocols. This is the Zeus Rete Engine and Rule-Processing module in Figure 2. This

module infers the degree of trust to assign a new principal on the basis of the set of trust profiles that the agent has received from other Phobos agents. This process is guided by the current security policy, which is communicated to the agent by its local master agent.

The *Acquaintance Database* describes the agent's relationships with other agents in the society and its beliefs about those agents' capabilities. The Coordination Engine uses information in this database when making collaborative arrangements with other agents.

The *Planner and Scheduler* plans the agent's tasks on the basis of the Coordination Engine's decisions and the resources and task specifications available to the agent.

The *Resource Database* maintains a list of resources (called *facts*) that are owned by and available to the agent. It also supports a direct interface to external systems, which lets it dynamically link to and use proprietary databases.

The *Ontology Database and Editor* stores the logical definition of each fact type—its legal attributes, the range of legal values for each attribute, any constraints between attribute values, and any relationships between the attributes of the fact and other facts.

The *Task/Plan Database* provides logical descriptions of planning operators (or tasks) known to the agent.

Finally, the *Execution Monitor* maintains the agent's internal clock and starts, stops, and monitors tasks that the Planner and Scheduler has scheduled for execution or termination. It also informs the Planner and Scheduler of successful and exceptional terminating conditions of the tasks it is monitoring.

I considered several alternative Java agent platforms during the design phase—for example, Cougaar (www.cougaar.org), a sophisticated open-source platform that DARPA developed, and the Agent Development Kit from Tryllian.com. Cougaar has a powerful distributed blackboard reasoning system and a clear plugable architecture. However, it is a heavyweight package with a substantial learning curve. The Tryllian ADK is a well-developed agent toolkit, with built-in P2P messaging capabilities, but it has licensing restrictions and focuses on mobile-agent applications.

Security Service

A separate code package called the Security Service contains the essential security functions that agents request. That is, the Security Service can

- Generate new certificates
- Revoke certificates
- Check certificates' validity and life span
- Encrypt data
- Generate public-private key pairs
- Validate principals' authentication details

When a new user logs into the system, the Security Service creates a new secure user object and writes this to an object database. This object contains a digitally signed X.509 certificate, a new strong password, and any requests the user has made. The Security Service also provides separate GUI components to allow administrative control of the certificates held in its keystore. In addition, each agent possesses a unique X.509 certifi-

Clearly, it would be useful if the agents could query alternative online data sources to acquire data on a user's history and the level of trust other organizations or systems granted.

cate to use during authentication exchanges with other agents.

The Security Service is also available to any resource applications that are plugged into the Phobos messaging network and can authenticate users requesting access to those resources. It also lets agents perform security checks on other agents in the system by validating the certificates presented by an agent whenever it initiates a new communication with another agent.

Current implementation

The target services for Phobos are any suitable distributed or Web-based service that requires strong user authentication. The current implementation is a Web-based computer shopping database accessed via a Java servlet interface. The servlet uses an instance of the Security Service to perform basic user authentication and security checks. The service can access either a remote copy of the authorized user database (maintained by the agent network) or a local replicated copy. The

servlet uses the Tomcat platform, which provides a stable, easy-to-install servlet engine.

The Phobos database component contains generic code to access three separate databases. The first is a full Structured Query Language Microsoft Access-format database, which uses local JDBMS (Java Database Management System) drivers. This holds the product catalog that the shopping-catalog servlet uses. A second Access database holds the messages generated between any plug-ins or agents. In addition, a flat-file object database contains serialized Java user objects. (We plan to migrate this to an XML-based relational database model.)

Future work

Several channels exist for further development of this work. First, it might be beneficial to let agents specialize so that they can become experts at particular aspects of system security or authentication—for example, by gathering background data on a user's trust status.

In the future, the message format will incorporate improved SOAP (Simple Object Access Protocol) XML integration to allow smooth interaction with standard Web services. Clearly, it would be useful if the agents could query alternative online data sources to acquire data on a user's history and the level of trust other organizations or systems granted—for example, by performing credit checks or looking up career history, if relevant. Because this raises privacy concerns, it needs careful policy implementation.

We are also conducting closely related research into methods for enabling groups of security agents to monitor the network's health via inhibitory messages between agents. In particular, if an agent fails to receive correctly signed messages from another agent, the system assumes the second agent is compromised and isolates it from the rest of the system.¹² In addition, we could group agents within restricted network domains to limit the potential damage due to a rogue or hacked agent.

A further advantage of this design is that plugging custom components into the underlying P2P message layer can easily extend the system. This can then provide additional security functions for the agents—for example, intrusion detection systems, network and resource monitoring, and monitoring users once they are granted access. Such a process also lets administrators implement an integrated security policy, with the network defense systems able to communicate automatically with the user-authentication agents.

A final issue for investigation is how to develop the hierarchy of management and delegation rights in the agent network. In the current version, one human administrator controls a team of agents in an intranet domain, which limits the system's scope for operating over larger network domains. Ideally, corporate-level administration of the agents should be possible with a hierarchy of agent delegation rights.

Using software agents to manage access and authentication in distributed networks appears to offer numerous advantages, specifically in reducing the operational costs for administering such services. However, several standard commercial security solutions aim to provide equivalent functionality—for example, SecurID (www.rsasecurity.com/products/secuid/index.html). Whether the agent approach is commercially successful will depend on whether the cost reduction warrants the investment by the major security vendors. However, the cost impact might shift in favor of the agent approach if it is applied in larger pervasive forms of networks where centralized user management becomes increasingly difficult. ■

Acknowledgments

Phobos is an internal project sponsored by the BTextact Technologies Intelligent Systems Laboratory. All material relating to the Phobos agent platform is the subject of UK and international patents.

References

1. A. Briney, "Security Focused," *Information Security*, Sept. 2000, pp. 40–68; www.infosecuritymag.com.
2. L. Kagal, T. Finin, and Y. Peng, "A Framework for Distributed Trust Management," *Proc. IJCAI-01 Workshop Autonomy, Delegation and Control*, 2001.

3. H.C. Wong and K.P. Sycara, "Adding Security and Trust to Multiagent Systems," *J. Applied Artificial Intelligence*, vol. 14, no. 9, 2000, pp. 927–941.
4. P. Maes, R. Guttman, and A. Moukas, "Agents That Buy and Sell: Transforming Commerce as We Know It," *Comm. ACM*, vol. 42, no. 3, Mar. 1999, pp. 81–91.
5. O. Etzioni, "Moving up the Information Food Chain: Deploying Softbots on the World Wide Web," *Proc. 13th Nat'l Conf. Artificial Intelligence (AAAI 96)*, vol. 2, AAAI Press, 1996, pp. 1322–1326.
6. M. Crosbie and E. Spafford, "Defending a Computer System Using Autonomous Agents," *Proc. 18th Nat'l Information Systems Security Conf.*, 1995.
7. G.G. Helmer et al., "Intelligent Agents for Intrusion Detection," *Proc. IEEE Information Technology Conf.*, IEEE Press, 1998, pp. 121–124.
8. D.M. Chess, "Security Issues in Mobile Code Systems," *Mobile Agents and Security*, G. Vigna, ed., LNCS 1490, Springer-Verlag, 1998, pp.1–14.
9. E. Bonsma, "Fully Decentralised, Scalable Look-Up in a Network of Peers Using Small World Networks," *Proc. 6th World Multi Conf. Systemics, Cybernetics and Informatics (SCI 2002)*, 2002, pp.147–152.
10. R. Rivest and B. Lampson, *Cryptography and Information Security Group Research Project: A Simple Distributed Security Infrastructure (SDSI)*, Laboratory for Computer Science, Massachusetts Inst. of Technology, 2002, <http://theory.lcs.mit.edu/~cis/sdsi.html>.
11. N. Minar, *Distributed Systems Topologies: Part 1*, O'Reilly & Associates, 2001, www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html.
12. F. Saffre, *ADDICT—An Automated Adaptive Firewall Algorithm*, BTextact Technologies, 2002.

For more on this or any other computing topic, see our Digital Library at <http://computer.org/publications/dlib>.

The Author



Robert Ghanea-Hercock is a principal research engineer at BTextact Technologies UK. He is an experienced researcher and lecturer in a wide range of AI systems, including software agents, mobile robots, and evolutionary algorithms. He has a BSc in physics and electronic engineering from Keele University, an MSc in real-time electronic systems from Bradford University, and a PhD in multiagent and robot control systems from Salford University. Contact him at BTextact Technologies, Main Lab Block, 1st Floor, pp12, Adastral Park, Martlesham Heath, Ipswich IP5 3RE, UK; robert.ghanea-hercock@bt.com or www.cybernetics.org.uk.



IEEE Pervasive Computing

delivers the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing and acts as a catalyst for realizing the vision of pervasive (or ubiquitous) computing, described by Mark Weiser nearly a decade ago.

In 2003, look for articles on

- Security & Privacy
- The Human Experience
- Building Systems that Deal with Uncertainty
- Sensor and Actuator Networks

To subscribe, visit

<http://computer.org/subscribe>

or contact our Customer Service department:

+1 800 272 6657
toll-free in the US and Canada
+1 714 821 8380 phone
+1 714 821 4641 fax

