

Securing Mobile Agent Based Tele-Assistance Systems

Sergio Pozo, Rafael. M. Gasca, and María Teresa Gómez

Department of Computer Languages and Systems
Faculty of Computer Engineering, University of Seville
Avda. Reina Mercedes S/N, 41012 Seville, Spain
sergiopozo@us.es, {gasca,mayte}@lsi.us.es
<http://www.lsi.us.es/~quivir>

Abstract. Nowadays the scientific community is trying to design new techniques in the search for solving security problems in mobile agent technology. There are now some industry initiatives for using agents in real environments which need a solution for some of their security problems. Companies offering assistive service are getting cost reduction due to tele-assistive technologies. In this paper we present a proposal based on secure tunnels to add confidentiality and integrity to any agent platform, present or future, without making any changes to its source-code. Our system is currently implemented and working as part of a complete mobile multi-agent system used for tele-assistance. During our research we have also detected many other problems regarding the tunnelling approach.

1 A new perspective of security in mobile multi-agent systems

There are a growing number of elderly people in Europe. Tele-assistive technologies are getting a lot of importance in cost reduction for companies and public insurance. Elderly community is also benefiting from a new era of comfort and additional services, such as continuous monitoring, full-time contact with relatives, new diversion services, etc. Relatives are also benefiting from this new kind of tele-assistance that lets them be in close contact with insurance centers and their family.

As tele-assistive or tele-care technologies cope with personal data, there are even specific laws about this data treatment. Depending on the data sensitiveness stored and treated by the companies (public or private) they need to use different protection technologies.

A relatively new approach to design and implement tele-care services is to use mobile multi-agent technology. Different classes of agents may be used to perform a large number of local and remote assistance and supervision tasks, such as sensorial data collection, alarm notification, health conditions monitoring, etc.

This new paradigm has a lot of practically unresolved security-related problems. Systems and users are exposed to access policy violations, communications integrity and/or confidentiality, system crashes, information theft, information modifications, etc. These problems and others added from the use of mobile agent technologies

needs to be resolved before any mobile multi-agent system could be used for a tele-care system, etc.

Since the very beginning of our research, our goal has been to provide a generic method for the protection of agents and agent platforms in a production environment. This solution needs to be deployable in current systems without further delays or new research and with a minimal effort.

In this paper we present an alternative and working way of protecting agents and platforms using existing tools and technologies in order to create secure tunnels. The work is structured as follows: first we present a very brief introduction to current threats of platforms and mobile agents and current theoretical solutions to the problem of mobility vs. security. Then we indicate the goals for our system architecture. Finally we propose a secure tunnel alternative to securing communications. We finish the paper indicating the conclusions and future work in this research line.

2 Current threats in mobile agent technology

There are several security threats in the agent paradigm (for both mobile and static agents), which allows at least two classifications [1], depending on the type of threat and on the entity of an agent environment which carries out an attack [1]. We have stressed the importance of security implications in a tele-care environment.

The most secure location for an agent is its home platform. Although neither agents nor home platforms are invulnerable, a number of conventional techniques can be applied to construct adequate defenses. Each time an agent migrates, there are security risks, and so it is needed a way to transmit this trusted environment to other platforms to which agents may travel. The greatest problem with multi-hop MAS is just the trust relationship which can be established in single-hop MAS between two platforms thanks to the security mechanisms derived from Client/Server architecture. These trust relationships are not transitive, nor have they to be bilateral.

2.1. General considerations

There are some practical solutions for securing communications in multi-hop MAS, but the vast majority of them include restrictions on itineraries. There are other alternatives based on the usage of a transport level protocol (usually SSL) to secure communications, and it has been applied with success to some agent platforms [2]. However, some of them are static MAS [3] and all solutions are directed towards a specific MAS. For tele-care it is need a mobile MAS and no itinerary-restriction.

The design and implementation of the agent platform itself is also of great importance. Java is a powerful language for implementing error-free MAS, but also the design phase is decisive in an application security development cycle.

2.2 Securing platforms and agents

Existing protection ongoing researches can be divided into solutions for protecting platforms and solutions for protecting agents [1].

Software-Based Fault Isolation [5], Safe Code Interpretation, State Appraisal [6], Path Histories [7, 8], Proof Carrying Code [9] are current research lines for protecting platforms and Partial Result Encapsulation, Execution Tracing [10], Environmental Key Generation [11], Computing with Encrypted Functions [12], Obfuscated Code [13] are for protecting agents.

3 Security goals for the architecture

We propose to divide protection techniques into three categories: protecting platforms, protecting agents and protecting communications. Our work is focused in the third category, providing no direct protection to platforms or agents themselves. Our solution provides protection to the communication channel and platform authentication between platforms. Our system also indirectly provides protection to platforms, forming a secure agent community.

We were looking for a solution that provides integrity, confidentiality, data origin authentication, mobile multi-agent system independence, compliance with standards, existence of cryptographic acceleration hardware and cost-saving and reuse of existing infrastructure.

Agllets [14] have been used for our reference implementation in the lab, since it is Open Source, multiplatform, easy to develop, has strong mobility (maximum mobility in a Java-coded MAS), high acceptance and relatively good documentation [15].

4 TLS, SSH2 and proprietary-protocol secure tunnels

Tunnelling is the capability of encapsulating one protocol within another, using this second protocol to traverse network nodes. A secure tunnel is one which encapsulates an insecure protocol (like FTP or HTTP) within a secure one (like SSL). Tunnels may also be used to bypass firewalls and are vulnerable to denial of service attacks, since they use a public and untrusted network as transmission media.

Due to the independence which the secure communications system must have from any mobile MAS, research has focused on systems which generate application-independent secure tunnels.

4.1. Stunnel

Stunnel [16] is an application which acts as SSLv3 server and/or client, providing a secure SSL-based secure tunnel (wrapper) for insecure protocols or applications with the only need of the installation of the application in each of the systems that needs to

secure. Stunnel is distributed under a GPL license and has versions for Microsoft Windows, some flavors of UNIX and many other OS. Stunnel also supports cryptographic accelerator hardware and client and server authentication with X.509 digital certificates (as SSL does).

The way Stunnel works is wrapping ports on the client and server systems (Fig. 1). It wraps the port the real service has assigned at the server (e.g. 4000 in Aglets server to another port, say 4001), and at clients it creates a local port the client service has to communicate with to access the Aglets network service (say 4001). The client connects to its 4001 local port. Then Stunnel wraps the connection and redirects the request to the 4001 port on the server node, which is also a port wrapped by Stunnel. The server-side Stunnel unwraps and redirects data to Aglets platform on port 4000 (which is usually blocked by a firewall to prevent direct external access). Authentication (server and optionally client) also takes place in the process at the beginning of the communication (like in whatever other SSL connection).

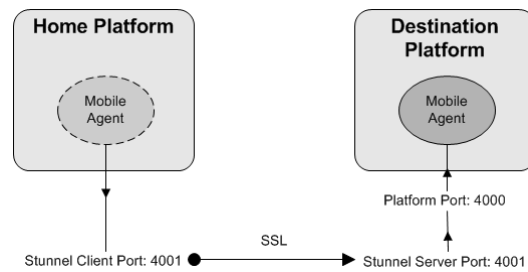


Fig. 1. Example Stunnel port-wrapping mechanism

In the case of a mobile MAS, the server and client ports are the ones where the mobile agent platform resides, although in the agent paradigm does not exist the concept of server and client, we will use them to refer to the platform where the agent dispatches, that is the source or client platform, and the platform where the agent moves to (destination or server platform). These roles of the platforms are always changing in multi-hop mobile MAS, and are inverted from the natural ones of a typical Client/Server architecture (Fig. 1)

A typical mobile multi-agent community can be modeled as some nodes that act as home platforms for the vast majority of agents, and other nodes that play the role of intermediate or visit nodes. The number of visit nodes is usually far bigger than the number of home nodes in a typical multi-agent community. Although any node can create an agent and become a home node, in real applications with multi-agent systems, the community is usually divided into platforms that create the majority of agents (sources) and other platforms that doesn't create mobile agents (destinations).

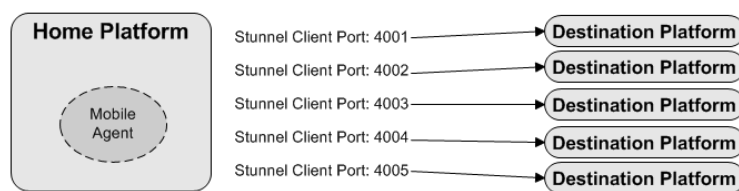
We have detected some problems regarding Stunnel during our research with secure tunnels. Stunnel features are summarized in Table 1. The most important ones are as well discussed.

Table 1. Stunnel features

Feature	Explanation	Can be solved?
UDP tunneling	Stunnel cannot wrap UDP protocol.	No
Node authentication	Stunnel uses digital certificates. A lot of certificates need to be issued.	N/A
Integrity checking	As Stunnel uses SSL protocol, it also uses the same mechanisms of data integrity.	N/A
Port-range mapping	Stunnel cannot map a range of ports in a unique process.	No
Tunnel dodging	Problems with Stunnel and all protocol wrappers in general.	Yes
IP address routing	When using IP addresses as basic authentication mechanism and NAT servers.	Yes

- Node authentication. A feature derived from using SSL is the need to use digital certificates for node-authentication. At least a digital certificate needs to be used in each destination platform. Client-authentication (source nodes) is optional in SSL.

Another authentication problem is derived from the fact that the role of server and client nodes are inverted from the natural ones of a typical Client/Server architecture, where the lesser number of nodes are the servers (and are the only ones that really needs digital certificates). With the Stunnel approach it is needed to create at least a digital certificate for each destination node of a community; furthermore as the role of each platform is always changing in multi-hop mobile MAS, each node needs a digital certificate. That is not a big problem with small communities, but with bigger or small but easily growing ones, it could be a strong restriction, because we must maintain a community-restricted PKI, and deploy strong authentication mechanisms for nodes to prevent tampering of certificates. A beneficial collateral effect derived from the deployment of strong authentication systems and PKI infrastructure is that no agent coming from external parties can be injected in our community. It is a natural isolation mechanism.

**Fig. 2.** Multiple destination Stunnel setup

- Port mapping. Stunnel communication model have some problems regarding the way it maps ports. Stunnel needs to map separate ports with separate IP addresses for each destination platform in order to be able to establish the tunnels (that is, it cannot map a range of ports or a range of IP addresses). One Stunnel process is needed on each source platform (5Mb) for each destination to which the agents

may travel to. On a network with 2000 possible destinations, the RAM required on the each source platform would become unfeasible (10,000 Mb of RAM, or 10Gb). It should be taken into account that the source platform is always client (Fig. 2.)

Another problem is that some agent platforms (such as Aglets) work on a fixed but user-configurable port for agent migration, but it uses a random port from a range for message transmission. Due to design restrictions of Stunnel, this actually means having one Stunnel process for each possible messaging port, which is totally unfeasible for the same reason as before (there could be 1000 ports in the range, and each Stunnel process requires 5Mb). Multiply this quantity of needed RAM to the one obtained from running one more Stunnel process for each possible destination and you could easily need a Terabyte-capable system to run an effective SSL-tunnelled mobile MAS in large communities.

- Tunnel dodging. Since Stunnel is external and transparent to the application which uses it, when an agent is asked for a handler or proxy in order to be able to reference it, the platform gives the address and port of the system where it is residing. It does not give the port where Stunnel is located, but the real platform port, and so the communication, if established, is going to be untrusted (bypassing Stunnel). Some agent platforms provide a proxy mechanism to encapsulate its transmission protocol in another protocol, creating a tunnel for it. This is the case of Aglets, where it is possible to define an HTTP proxy (host and port) to use when transferring agents. That is, Aglets encapsulates ATP over HTTP, and then HTTP requests are encapsulated in SSL through Stunnel and transmitted.
- IP address routing. There is a problem related to private IP addresses, NAT servers and some mobile MAS platforms (such as Aglets). This problem is not related to tunneling itself but to specific agent platform design itself. If the mobile MAS platforms have private IP addresses and are behind NAT servers, the agents need to traverse that NAT server, which is going to change the IP address of the packets to the one the NAT server has. Some mobile MAS hardcode the home platform IP of the home platform in the agent on dispatch. This address is then compared with the one of the packets arriving at the destination platform, and if it is not the same, the agent will be discarded. This is a general problem with NAT and could not be easily resolved because it depends on the design of each mobile MAS platform. Some mobile MAS tries to resolve the hardcoded IP address of dispatched agent, but if the address is in the private range it is not possible to resolve it unless there is an available DNS server that has registered that private IP.

4.2 Secure Shell (SSH2)

SSH is a protocol for securing network services over an insecure network. It is traditionally used for protecting insecure UNIX protocols such as telnet, rlogin, etc. Moreover SSH can be used to secure other services creating a wrapper around them using a local port redirection scheme very similar to the one used by Stunnel. SSH is widely accepted by the scientific community as being a trusted security protocol.

SSH architecture is very similar to SSL or TLS one, and provides basically the same functionality. SSH can use different authentication schemes such as preshared secret or digital signatures. It also uses Diffie-Hellman key agreement protocol (such

as TLS) and HMAC integrity hash. Different implementations of SSH protocol can use different encryption algorithms. SSH features are summarized in Table 2.

There are several implementations of SSH protocol, ranging from Open Source to commercial ones and for a wide variety of operating systems. Although very similar to Stunnel at protocol level and working way of OpenSSH implementation, this application has been considered for its wide acceptance.

Table 2. SSH (OpenSSH) implementation features

Feature	Explanation	Can be solved?
UDP tunneling	SSH doesn't support UDP.	No
Node authentication	Stunnel uses digital certificates. A lot of certificates need to be issued.	N/A
Integrity checking	SSH uses HMAC hashes.	N/A
Port-range mapping	OpenSSH cannot map a range of ports in a unique process.	No
Tunnel dodging	Problems with SSH and all protocol wrappers in general.	Yes
IP address routing	When using IP addresses as basic authentication mechanism and NAT servers.	Yes

4.3 Zebedee

Zebedee [18] is another Open Source application used to create secure tunnels with implementations in Windows, UNIX, Linux, Java and Ruby. Zebedee (from its documentation) has a small memory footprint and low wire protocol overhead.

Zebedee does not use SSL, but a plain Diffie-Hellman protocol for key agreement and a symmetric key cryptographic algorithm, Blowfish. Zebedee does not provide any features for data integrity.

In counterpart, Zebedee solves some of the design and implementation problems of Stunnel, providing us with a more flexible way of creating tunnels. In exchange we get a weak authentication mechanism and no integrity checking. Zebedee also supports compression (through zlib and bzip2 libraries) with a selectable range of speed and efficiency. It doesn't support cryptographic acceleration hardware because it isn't based on standards.

- Tunneling of UDP protocol. Zebedee can be used to tunnel UDP protocols, but even in UDP-mode the created secure tunnel uses TCP protocol to wrap it.
- Node authentication. Zebedee permits a sort of really weak identity checking mechanism for authentication (it is not possible to use digital certificates for authentication): one based on IP addresses, and the other based on the generation of a private and permanent key, that is used (along with the modulus and generator of the Diffie-Hellman phase) for the generation (using a hash function not specified) of a public fingerprint which needs to be copied in every Zebedee server it needs to communicate with.

As neither the private key nor the modulus and generator change, and although Zebedee uses session keys that change regularly, this identity checking scheme is vulnerable to known-plaintext attacks. Zebedee documentation also points that its identity checking mechanism is also vulnerable to man-in-the-middle attacks.

- Port mapping. Zebedee can also map a range of IP addresses and ports for destination platforms in only one process, thus releasing the huge RAM constraint of Stunnel on big agent communities and the random port selection when sending messages in some agent platforms, thus improving the maintenance of the architecture.

Table 3. Zebedee features

Feature	Explanation	Can be solved?
UDP tunneling	Zebedee uses TCP to wrap UDP protocol.	N/A
Node authentication	Really weak authentication scheme based on IP addresses.	No
Integrity checking	Zebedee does not support any integrity checking mechanism.	No
Port-range mapping	Zebedee can map a range of ports in a unique process.	N/A
Tunnel dodging	Problems with Zebedee and all protocol wrappers in general.	Yes
IP address routing	When using IP addresses as basic authentication mechanism and NAT servers.	Yes

In spite of its weak authentication mechanisms, this weakness can be used to simplify security maintenance, as it doesn't need any private secret to be permanently stored at any platform (it should be remembered that key generation is automatic). Adding nodes to the infrastructure is straightforward, simple and cost-effective, although very insecure. A PKI isn't needed this time, but the need of the DNS still remains.

The lack of standards supported by Zebedee, weak authentication mechanisms, lack of integrity checking mechanisms, and the impossibility to resolve private IP address routing make Zebedee a near-unusable alternative to Stunnel or SSH. Zebedee features are summarized in Table 3.

4.4 Security and availability problems with TCP-based secure tunnels

In addition to the exposed problems in all secure tunnel architectures, whether it is a SSL-based system or a proprietary one, there is another serious security and availability problem related to all TCP-based tunneling systems.

The capability of inferring TCP sequence numbers (commonly named blind spoofing attack) have been a fact for many years [19]. This can be used to inject packets in an open TCP conversation or to send connection termination packets to one or both parties, causing severe delays and in the end a DoS failure [17]. It is possible to carry out the attack injecting packets from a third party before tunnel establishment (although it is difficult to carry on a blind spoofing attack out of a LAN), since TCP-

based secure tunnels doesn't authenticate TCP packets until the tunnel has been established.

5 Conclusions and future research

Mobile multi-agent systems introduce a lot of security issues. Some of them are new and others are very similar to client-server paradigm ones. The majority of the solutions are directed towards a concrete mobile MAS or are rather theoretic. Mobile MAS applications can be used nowadays by the industry in tele-care services, since there are even commercial MAS platforms. But as programming languages are becoming multiplatform, it is clearly needed multi-platform security solutions for at least some of the security issues they have. A tele-care service is even more restrictive.

We propose in this paper a new approach for protecting agent communications using secure tunnels implemented using existing applications and protocols. We were looking for a cheap, multi-platform and generic solution to several mobile MAS platforms. This solution is easily applicable to any service, including tele-care ones.

Our work is focused on protecting communications, but also provides some collateral effects that improve the protection of agents and platforms. Using a key distribution scheme (i.e. a PKI) for our community, guarantees that only agent platforms whose digital certificate have been signed by the certificate authority of the community PKI can communicate with other platforms of the community.

Our solution provides privacy, integrity checking and a basic form of authentication scheme that needs to be extended to be usable in a large production environment. Our solution is also mobile MAS independent and very low cost. Existing tele-care agent applications can obtain a direct benefit with very few or even no modifications at all.

We have studied in depth the solutions that secure tunnel applications can provide to existing agent platforms, using Aglets as the reference platform. During the study new non-security related problems have also been detected (for example, routing of IP addresses and authentication scheme of some agent platforms). Some of them have been solved, but others are limitations of the applications that have been tested. There are some other applications we have not tested (such as SSLWrap) because they provide the same functionality to the ones that have been tested. We kept the most representative and actual ones.

Since our solution is a preventive technology, it cannot provide predictive or detection capabilities on its own. Neither human access control nor authentication has been considered, since these issues are complex enough to dedicate its own research efforts. These lines are areas for future research. There is also some room to research in order to provide solutions to the new problems created by these secure wrappers, to the ones that still remain after our solution, and finally for performance testing of the proposed solutions.

References

1. W. Jansen, T. Karygiannis: Mobile Agent Security. NIST Special Publication 800-19, October 1999.
2. L. M. Silva, P. Simoes, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida, N. Stohr: JAMES: A Platform for Mobile Agents for the Management of Telecommunication Networks. Proceedings Intelligent Agents for Telecommunication Applications, IATA'99, Stockholm, Sweden, August 1999.
3. A. Puliafito, O. Tomarchio: Design and Development of a Practical Security Model for Mobile Agent System. 7th IEEE International Symposium on Computers and Communications, ISCC'02.
4. S. Fischmeister, G. Vigna, R. A. Kemmerer : Evaluating the Security of Three Java-Based Mobile Agent Systems. 5th IEEE International Conference on Mobile Agents (MA'01). Atlanta, Georgia, USA, December 2001.
5. R. Wahbe, S. Lucco, T. Anderson: Efficient Software-Based Fault Isolation, October 1998.
6. W. Farmer, J. Guttman, V. Swarup: Security for Mobile Agents: Authentication and State Appraisal. Proceedings of the 4th European Symposium on Research in Computer Security (ESORICS), September 1996.
7. J. J. Ordille: When Agents Roam, Who Can You Trust? Proceedings of the First Conference on Emerging Technologies and Applications in Communications, May 1996.
8. D. Chess, B. Grosz, C. Harrison, D. Levine, C. Parris, G. Tsudik: Itinerant Agents for Mobile Computing. IEEE Personal Communications, vol.2, no. 5, October 1995.
9. G. Necula, P. Lee: Safe Kernel Extensions Without Run-Time Checking. Proceedings of the 2nd Symposium on Operating System Design and Implementation (OSDI), October 1996.
10. G. Vigna: Protecting Mobile Agents Through Tracing. Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, June 1997.
11. J. Riordan, B. Schneier: Environmental Key Generation Towards Clueless Agents. Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
12. T. Sander, C. Tschudin: Protecting Mobile Agents Against Malicious Hosts. Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
13. F. Hohl: Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
14. D. B. Lange, M. Oshima: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, 1998. ISBN 0-201-32582-9.
15. J. Altmann, F. Gruber, L. Klug, W. Stockner, E. Weippl: Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms. 5th International Conference on Autonomous Agents, 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents. Montreal, Canada, May 2001.
16. W. Wong: Stunnel: SSLing Internet Services Easily. SANS Institute, November 2001.
17. A. Pankratov: Trivial Denial of Service Attack against TCP-based VPN. Security Focus, April 2003.
18. N. Rinsema: Secure (and free) IP Tunneling using Zebedee. SANS Institute, June 2001.
19. M. Zalewski: Strange Attractors and TCP/IP Sequence Number Analysis. BlindView Corporation, 2001.