

AMETAS

Good Migrations.

AMETAS White Paper Series

by Klaus Herrmann
and Michael Zapf

On Security

July 2000

Johann Wolfgang Goethe-Universität Frankfurt/Main, Germany
Department of Computer Science

www.ametas.de

In Our System, You Feel Comfortable And Secure¹.

Security is a crucial aspect when using an application, and not only when you do your e-banking. If there are any holes in your security protection, attackers may gain access to your system and cause considerable damage to your system and probably to you as well, especially when they are able to disclose private data. In the early days, agent systems were designed with the ease of use in mind, but nowadays it has become clear that a system with no security facilities is not viable in serious application areas.

I Have Nothing to Hide.

You might tend to believe that. Many users do — usually the same ones that cannot imagine to write all their correspondence on postcards to be delivered by totally unknown people. Well, as for your e-mail messages, you probably really don't care if anybody else reads them. But if you are using an agent system you should *very quickly* change your mind.

Starting from that moment when you start your agent place on your host, you invite anyone to send code to your host that is at best unknown and at worst malicious. Unlike applets, not *you* decide when code comes to your machine but everyone else that can get in contact with you over the Internet. Don't we have yet another good model of worms here?

If anyone promises you security it is a good advice to start with disbelief. Ensuring security is a *really* tough job, and we actually do not know any system where it is harder than with an agent system. Therefore, the security subsystem of AMETAS was very carefully designed. More than half of the time of development of the core system was dedicated to the development of the security subsystem. More than half of all classes of the AMETAS distribution are directly or indirectly related to security issues.

In the following sections we describe the basic facilities of the security system and hope to convince you that most of the probably security attacks can be effectively defeated by the AMETAS security subsystem.

Attacking Your Agent System

An agent system consists of several parts, among those are the places, the Place Users, the users, and messages that are exchanged between the components. The intentions of an attacker can be manifold:

- The attacker tries to grab your agent to inspect the data in it. If it carries confidential data with it (e.g. your credit card number), the attacker may be able to extract it and misuse it.
- The attacker tries to replace your agent with an own version that contains goals in favor of the attacker, unnoticed by you.
- The attacker attempts to disturb the transmission of the agent or other data.
- The attacker sends an agent to your place that tries to gather confidential data or even attack your complete system.

¹And Earth is flat and cows can fly.

- The attacker provides an agent place and invites you to send your agents to him. His places, however, try to inspect your agents and probably change their behavior completely.

This is only a short collection. Note that in contrast to all previously known dangers, the aspect of having code wandering between the nodes effectively adds a new dimension of security attacks.

Code Integrity

One important requirement for a secure system is that the integrity of the data is preserved. In agent systems, you would want to be sure that your agents' code is unmodified during the whole time of using the agent. This is ensured by using *signatures*.

AMETAS Place Users are packaged in so-called *Signed Place User containers* or short *SPUs* (Figure 1). A SPU contains all classes that belong to the agent; when it migrates, the state of the agent together with the corresponding SPU of the agent is transferred.

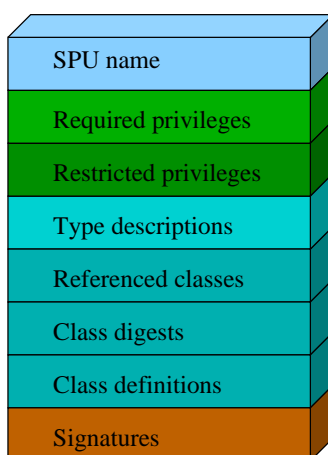


Figure 1: Signed Place User container

It is important to note that it is impossible to modify this SPU unnoticed: When the agent is loaded and instantiated the first time, the SPU container is automatically signed on behalf of the user. This signature covers the agent code, the privileges, and the agent type description. Signatures are created using the MD5/RSA signature algorithm. To be sure that the agent code is unmodified before it is started the SPU contains author signatures. These signatures prevent any change in the class code and any replacement of classes.

Privacy

Privacy is the state when you are sure that no other person may find out what your plans are when employing agents. How can this privacy be enforced?

At first it must be impossible for the attacker to *eavesdrop* the communication between the places. This is achieved by using encrypted communication. Furthermore,

to prevent the *Man-In-The-Middle* attack, agent places have to authenticate with each other using the Needham/Schroeder authentication protocol.

After the two places have checked that each one is the one it claims to be, they use the exchanged secret to establish a symmetrically encrypted channel. AMETAS implements such an algorithm as the symmetric cipher. This encryption also allows to check if the transmission was disturbed. In this case the code is rendered invalid and the migration fails, notifying the agent at the sender place about this incident. The agent may then try to repeat the migration. The procedure is shown in Figure 2.

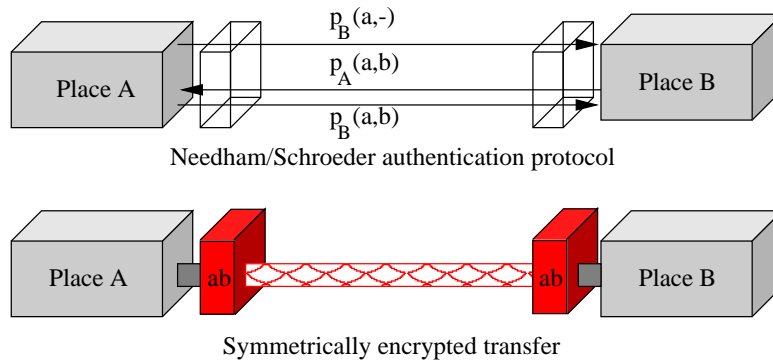


Figure 2: Ensuring transfer privacy

Agents in AMETAS are not directly accessible by another object except for the place that provides its execution environment. This effectively protects the internal state from being inspected by other Place Users. Furthermore, Place Users cannot intercept local transmissions between other Place Users because of this strict separation. Messages are distributed by a place-internal facility called Post Office which cannot be directly accessed by Place Users. They can only deposit and retrieve messages by predefined driver methods.

A security hole concerning privacy may be the graphical display of user adapters that can probably be inspected by exploiting misconfigured hosts or trying to capture compromising radiation from monitors. This is actually a general problem not directly related to agent systems.

As described in more detail in the paper about interfaces in AMETAS, the connection between the interfaces AMAI/AMSI and the place is also encrypted to protect your passphrase from being disclosed. Usually the passphrase never leaves the place you connect to. You may request the system to send your passphrase along with the agent (so that agents remotely started may be signed on your behalf). This is, however, only possible when the migration may always utilize encrypted channels. If any single channel is not encrypted the copy of the passphrase is discarded. Always keep in mind:

Never ever allow anyone to get the passphrase of your identity. If that happens, you must immediately discard your identity file and get in contact with the administrators of all places your identity is registered so that it can be deleted. Any action of the person in possession of your passphrase will be considered as *your* action.

Afraid of Worms And Viruses?

Of course, we do not mean the small animals but those that you hear of spreading like a plague on the Internet. One of the most important questions is: Can an agent come to your place, perform some actions, unwanted by you, and probably even get in control of your place, using it as a platform to break in your complete system?

AMETAS introduces the concept of privileges and permission. (For details see the corresponding paper in the *AMETAS White Paper Series*.) At first note that an agent coming to your place gets all permissions and privileges *from your place*, not from any remote entity. That means: Regardless of what permission the agent had where it departed, your place determines the set of permissions and privileges the agent will get.

This relies on the determination of the sender and authors of the agent. You may configure the security system to reject all agents that were created by unknown authors. The set of privileges is determined by finding out who is the sender and then checking the entries in the domain access policy of the current place. As a place administrator you always know the maximum set of privileges and permission an agent from a certain sender will get when coming to your place.

In everyday work you will probably find that a Place User has too *few* privileges instead of too many. But be careful not to grant any kind of privilege to any user just for the sake of comfort. Remember that you can provide services to do those jobs that an agent is not entitled to do.

The Least Expected Security Problem

Yes, there remains another problem, and this one is actually the hardest one. Imagine you want to take part in an electronic market and send out your buying agent to look for some books. Having returned you find out that your agent has not only spent all your cyber money but has also ordered quite some expensive cars, booked several flights and so on. Analyzing the agent does not help; what happened is simply impossible, at least when looking at the code. On the other hand, the owner of the other place can prove that it was exactly your agent who did this purchases.

Probably the only plausible explanation that remains is that the other place *attacked your agent during its stay at that place*. The place could have tried to modify internal fields of the agent — remember that the place is the only entity that bears a reference to the agent —, tried to modify the messages coming from your agent or do anything it liked to do instead of executing the code in the way you would expect it.

This is a security problem that was completely unknown before the new paradigm of mobile, autonomous agents arose. The question is: Can I make sure that the remote place reacts in a foreseeable way when I send my agent to it? Many researchers have been searching for a solution, presenting strategies like encrypting parts of the agent or having it executed in a special, certified portion of hardware.

It might be a bit premature to say, but there are strong arguments that support the fear that the ultimate security in this respect is simply *out of reach*. Too negative? Consider this: What do you know about the remote place? You only see its open socket. You send your agent as a stream of data to it and expect this black box to do something with this data. Then your agent returns again as a stream of data. Regardless what methods you try to apply, it always boils down to this: As long as you do not have control of the remote place directly, it stays a black box for you. Even if the owner of

it assures you that he uses trusted hardware, you cannot get a proof for this².

Of course, you can send encrypted data with your agent that can only be decrypted by the place that is intended to do this. However, whenever any portion of the agent is exposed to a place, this very portion is subject to an attack by this place. This is unavoidable. When people promise any security of your agents against a place, be skeptical.

Agents aren't that a good idea after all, you could now believe. What are agents good for if there is such a big security hole without any prospect of closing it in the near future? Well, we would suggest two things:

- If the application area is designed in a way that allows you to check all agent places for compliance to a unique behavior, the risk of attacking places can be minimized. Place Users are subject to principal restrictions that cannot even be broken in case they have all available permissions. That means if you can make sure that the place does not misbehave as such, the services running on that place should not be able to attack your agents as well.
- If the application area is open in the sense that you do not have access to all places, only have your agent do things that cannot go severely wrong. Do not send confidential data with your agents to places you do not know.

This might be the most important suggestion that applies as well in real life: *Avoid sinister places*. The aspect of *reputation* could become the best assurance for places to behave correctly. If places are known to cheat agents, the owner will sooner or later be forced to withdraw from the electronic market. Consider how, in everyday life, we deal with rogues: Once discovered, we chase and hopefully arrest them so that they will not be able to carry on with their vicious plans. Why should it be different in the virtual life of agents and places?

²We once considered to request the hash code of the remote Java archive file containing the complete AMETAS distribution. But who can tell for sure if that what we receive is derived from the code that is actually used for executing the place? How can we prove that the virtual machine that is used during the check is the same that is used to execute our agent?