Applying Mobile Agent Technology to Intrusion Detection

Christopher Krügel
chris@infosys.tuwien.ac.at

Thomas Toth
ttoth@infosys.tuwien.ac.at

Distributed Systems Group
Technical University Vienna
Argentinierstrasse 8
A-1040 Vienna, Austria
+43 1 58801 18451

ABSTRACT
The increasing number of network security related incidents makes it necessary for organizations to actively protect their sensitive data with the installation of intrusion detection systems (IDS). Autonomous software agents, especially when equipped with mobility, promise an interesting design approach for such applications.

We evaluate the implications of applying mobile agent technology to the field of intrusion detection and present a taxonomy to classify different architectures. Sparta, an actual implementation of a mobile agent based system which is developed at our group is described as well.

Keywords
Intrusion Detection, Network Security, Mobile Agents

1 INTRODUCTION

After the concept of intrusion detection (ID) was first established in 1980 by Anderson [1] and later refined by Denning in her famous article [6], two major variants of intrusion detection systems (IDS) have emerged, namely host and network based approaches. Host based systems collect local data from sources internal to a computer, usually at the OS level. This has the advantage of collecting high quality data directly at the source (e.g. kernel). Unfortunately, some attacks cannot be detected at a single location. Distributed intrusions may leave innocent marks at each single host and can only be identified when combining data from a number of different machines. In addition to that, worms or telnet chains (i.e. successive logins by an attacker to hide his tracks) are easier to spot when data from several sources is considered.

Network based variants monitor packets on the wire by setting the network interface to promiscuous mode and analyzing network traffic. Therefore they have some possibilities to correlate activities that occur at different hosts, but suffer from scalability problems in case of high network load and have problems when encrypted communication is used.

A new approach is the development of distributed architectures, where sensors (host and network based) collect data, preprocess it and send it to a centralized analyzing station which is able to relate this input. Such client-server architectures suffer from the following deficits.

- A central analyzer is a single point of failure. When an intruder manages to put it out of action (e.g. denial-of-service attack), the whole network loses its protection.
- When all information is processed at a single location, the system is not scalable. The processing capacity of the analyzer unit limits the monitored network size and distributed data collection can lead to excessive data traffic over the network.
- It is difficult to apply reconfigurations to the sensor stations. Usually, the whole system has to be restarted after a modification.

The classic solution to combat these shortcomings is the introduction of several hierarchical layers and redundant components. State-of-the-art ID systems like Emerald [13] or AAfID [3] use that approach.

A peer-to-peer intrusion detection system without a central processing station is proposed in [15]. All hosts run a local ID system and a security manager that can process input from the local host as well as from other security managers. These security managers cooperate by message passing to detect distributed attacks.

A different and interesting approach is taken by systems which utilize mobile agents to perform distributed intrusion detection. The aim of this paper is to discuss advantages and possible drawbacks when applying mobile agents to intrusion detection systems. In addition to that, we introduce a taxonomy which can be used to classify such ID systems and briefly present our implementation of an ID system called Sparta, which heavily relies on mobile agents.

2 MOBILE AGENTS

The development of distributed ID systems and the introduction of software agents to perform intrusion detection lead

to the idea of using mobile agents. Mobile agents offer several potential advantages when used in ID systems (see also [9]) that may overcome limitations that exist in IDS that only employ static, centralized components (as discussed above).

- *Reducing Network Load* - Instead of sending huge amounts of data (e.g. audit files) to the data processing unit, it might be simpler to move the processing algorithm (i.e. agent) to the data.
- *Overcoming Network Latency* - When agents operate directly on the host where an action has to be initiated, they can respond faster than a hierarchical IDS that has to communicate with a central coordinator located elsewhere on the network.
- *Autonomous Execution* - When portions of the IDS get destroyed or separated, it is important for the other components to remain functional. Independent mobile agents can still act and do useful work when their creating platform is unreachable which increases the fault-tolerance of the overall system.
- *Platform Independence* - The agent platform allows agents to travel in a heterogenous environment and inserts an OS independent layer between the hosts and the IDS using agents. This allows IDS to share data (and build a common knowledge base) as agents from one organization may visit other ones and collect data there (if allowed).
- *Dynamic Adaption* - The mobility of the agents can be used to reconfigure the system at run-time by having special agents move to a location where an attack currently takes place to collect additional data.
- *Static Adaption (Upgradability)* - It is important for an (especially misused based) IDS that its attack signature database and the detection algorithms are up-to-date. Instead of upgrading and restarting all sensors when new signatures are available, it is simpler to write updated agents and send them on duty while the IDS keeps running.
- *Scalability* - When a central processing unit is replaced by distributed mobile agents, the computational load is divided between different machines and the network load is reduced. This enhances scalability and additionally supports fault-resistant behavior.

Unfortunately, the introduction of agents and agent platforms may also cause the following problems [8].

- *Security* - Introducing agents into an IDS causes several security implications that must be considered. On one hand, the host (and the agent platform) where an agent gets executed must be protected against malicious code. This can be done by signing agent's code and providing a valid certificate, that can be checked by the platform. On the other hand, agents can be modified or be subject to an eavesdropping attack when they move over the network. This might be prevented by encrypting agents in transit. Additionally, mobile agents can be attacked by a malicious agent platform itself. This threat is extremely difficult to fight when agents need unrestricted movement around the network [7].
- *Code Size* - An IDS is a complex piece of software and agents that implement its functionality might get rather large. Transferring the agent's code over the network may take some time, but it is only needed once, when each host stores agent code locally. [8] claims that agents get especially large when they encode operating system dependant parts, but one might consider putting these routines into the agent platform and offer a generic interface to agents (effectively overcoming this drawback).
- *Performance* - Agents are often written in scripting or interpreted languages to be easily ported between different platforms. This mode of execution is very slow compared to native code. As an IDS has to process a large amount of data under very demanding timing constraints (near real-time), the use of MAs could degrade its performance.

An important aspect of an IDS is its capability to detect multi-point attacks. It is a question under research, how agents can efficiently collect and analyze only parts of the data. Instead of moving huge amounts of data to a central point, the idea is to utilize mobile agents. They can work in a collective fashion without simply carrying the whole data with them (which would in fact increase network load). Our proposed solution to this problem (that has been implemented in Sparta) is described in Section 4.

Additionally, agent systems could be used to create more attack resistant architectures. When a couple of aggregating processing units are replaced by a fully distributed agent system, the number of weak points is reduced. Agents act like a colony of insects when working towards a common goal. Nevertheless, it remains an open question how a global system view can be established by independent agents without a central coordinator.

Agents can be seen as guards, which protect a network by moving from host to host and performing random sampling. Instead of monitoring each host at any time, agents only visit machines from time to time to conduct their examinations. When any anomaly is detected, a more comprehensive search is initiated. Although the metaphor of patrolling guards seems appealing at first, this approach has the disadvantage of leaving hosts vulnerable while no agents are present. On the other hand, random sampling definitely reduces the average computational load at each machine.

3 CLASSIFICATION

A number of taxonomies for intrusion detection systems as well as for agent systems (e.g. [12]) exist. Nevertheless,

the application of agents in the context of intrusion detection raises issues which aren't sufficiently dealt with in either classification. Therefore, we have created a new taxonomy model for ID systems that utilize agents.

The following points are of interest for our model.

- *Agent Tasks* - Agent based intrusion detection systems utilize agents for different tasks. One can imagine a number of different designs, where agents fulfill parts or the whole functionality of the ID system. The basic functional parts of an IDS are (according to [10]) data gathering, data processing, data storage and response components. Data gathering is needed to collect information from various sensors to get a picture of the system state. Data processing is necessary to extract potential attacks from the often enormous amounts of raw data delivered by the data gathering unit and the response component is used to react on an intrusion (e.g. notify an administrator, reconfigure the firewall). The data storage serves as a persistent storage for collected data for later analysis and correlation.

  All of these parts can be implemented by agents, where most of the time, at least data gathering and processing is realized directly by them. It is interesting to know for a system whether it realizes data gathering, data processing, data storage or response components as agents.
- *Attack Description* - An interesting characteristic of agent based IDS is the way that users may specify intrusion scenarios which should be discovered by agents. Three possible ways can be identified. A common way is to implicitly describe attacks by providing code that directly operates on data structures delivered by data gathering components. The code itself determines whether an intrusion has occurred by processing the input and calling appropriate response functions.

  Another possible way that separates ID systems into components is the specification of scenarios in an application-specific scripting language. Usually, one is supported by predefined data types (e.g. IP packets) or a rudimentary way of expressing timing constraints.

  The last approach is a special language which allows the security officer to define attack patterns which consist of a set of events that can be spatially and temporally related. The description of the attack is translated into rules and code, which can directly be processed by agents. This has the advantage of an intuitive description of the attack scenario.

  The shown description methods are listed in increasing order of expressiveness. For each system, the kind of used language is an important design decision. A simple language might be easier to implement, but it can fail when new, distributed scenarios should be described. When choosing an extensive language the system design gets more complicated.
- *Data Relation* - A challenging issue when building an IDS is its mechanism to relate information from different sources. A common way is to rely on a client-server architecture, where client nodes forward their sensor information to a central analyzer component. This approach suffers from scalability problems when the number of clients grows and introduces a single point of failure into the system. Standard approaches like replication or additional hierarchical layers mitigate the negative impacts but do not attack the core problem itself. An alternative variant is a fully distributed design, where all nodes are equal and form temporary groups to detect distributed attacks. This approach offers improved fault-tolerance and scalability, but suffers from difficulties when coordinating the cooperating hosts.

  Systems which use central entities have to attack totally different issues than distributed ones. A design with a central processing unit has to deal with the weaknesses of a single point of failure and faces potential performance and scalability problems. Distributed systems on the other hand have to solve the problems of ordering events from different sources (time synchronization) or coordinating and locating agent activities.

  Another issue is the cooperation and communication between agents themselves. When agents can work together and in parallel to solve a common task, a system scales better than one, where a single agent has to visit every interesting host sequentially.
- *Persistency* - Two different types of agents are possible. One type (called transient) is launched by a central station or by another agent to perform a single, specific task. After results have been delivered, the agent itself vanishes. Such agents are mainly used for data gathering tasks. The other type (called persistent) are agents, that remain active for a longer period of time. Such agents have the ability to accumulate knowledge over time and react differently to identical stimuli. They roam the network on their own initiative and usually follow a broader range of activities. They can be considered as permanent guards that hop from host to host to perform checks.

  It is important to distinguish between these agent types because persistent agents have the ability to modify their reactions throughout their lifetime. Systems utilizing persistent agents can adopt to certain threat scenarios while transient agents usually cannot.

4 INTRUSION DETECTION SYSTEMS

Only a few research projects have already attempted to incorporate some ideas of mobile agent technology into intrusion detection systems. One called Micael [5] aims to realize the entire system functionality with mobile agents. Although the architectural description is interesting no implementation has been provided so far. Recently, another system which

is based on mobile agents has been introduced in [4]. Unfortunately, only an design overview is presented while the actual detection and correlation mechanism is dealt with superficially. IDA [2] is a classic host based system which uses agents to track attackers and perform active response (i.e. counterattacks). A few other systems, which claim to use agents in some way do not really fit our area as those agents are static. The most known one is AAfID [3], an architecture where distributed sensors are called autonomous agents. These agents cooperate in a client-server fashion by sending data to central stations where it is further processed. Although mobility is not an issue AAfID is frequently referenced in the descriptions of the systems mentioned above. Several other agent based systems without mobility are listed in [8].

The following subsection briefly describes Sparta, the system which is currently developed at our group. It should act as a proof of concept which demonstrates that the potential advantages of mobile agents for ID systems can actually be realized.

Sparta
Sparta [11] (which is an acronym for Security Policy Adaptation Reinforced Through Agents) is the name of a project sponsored by the European Union. It is a system whose primary aim is to detect security violations in a heterogenous, networked environment. Nevertheless, the architecture which has been designed for Sparta targets a broader range of applications, ranging from network management to intrusion detection.

Sparta is an architectural framework which helps to identify and relate interesting events that may occur at different hosts of a network. In addition to the detection of interesting patterns, Sparta can also be utilized to collect statistical data (i.e. extreme value or sum of attribute values) of certain events.

The system monitors local events at a number of hosts connected by a network. In order to deal with complex patterns, it is not sufficient to select events based on content alone. It is necessary to consider multiple events at the same time and deduce knowledge that is beyond the scope of an individual event (a process called *correlation*).

Each host has at least a local event generator, a storage component and the mobile agent platform installed. The local event generation is done by sensors which monitor interesting occurrences on the network or at the host itself. The exact types of events and their attributes is determined by the application's needs. In the current setup, we use Snort [14] to extract interesting events from network traffic. The events are stored in a local databases for later retrieval by agents. The mobile agent subsystem is responsible for providing a communication system to move the state and the code of agents between different hosts and for providing an execution environment for them. Additionally, the system has to provide protection against security risks involved when utilizing mobile code. Most of the components are written in Java and the agent platform itself rests on Gypsy [12], a Java based system which has been developed at Technical University Vienna.

The goal of Sparta is the design of a mobile-agent based IDS that identifies and improves potential shortcomings of other intrusion detection system designs. The following three issues are addressed.

To support our detection algorithm and to address the problem of systems which only offer an implicit way of specifying attack scenarios, we have designed an attack pattern language (called EQL). This language allows us to express offending event correlations in a declarative manner where one can specify what to detect instead of how to detect. The primary language design objective is the reduction of the needed amount of transferred data while still retaining enough expressiveness to be usable for most situations. When a system uses mobile code (i.e. mobile agents), it should aim at performing flexible computation remotely at the location where the interesting data is stored. This resulted in the limitation of restricting patterns to tree-like structures when events between nodes are involved. For events that occur at a single node virtually arbitrary correlations are allowed. In addition to the specification of patterns, EQL also allows us to define simple statistical queries. These can deliver the number of authentication failures of a certain user or identify the maximum number of processes running at a single machine of the monitored network.

We realized a correlation mechanism which does not rely on (one or more) central server locations where data is gathered and related. Instead, it follows a fully distributed approach. Mobile agents locally select interesting information and only move parts of the data across the network. When detecting patterns agents first try to find actual events which are specified by (and match) the root node of a given tree pattern. When the root node is located, the agents follow the branches of the tree to detect events that match the root's predecessors. This process is recursively applied until the whole tree has been matched. With this algorithm only very few data has to be carried by agents during each hop (for a complete description of the detection process refer to [11]). The detection algorithm is performed by multiple agents in parallel which improves scalability, fault tolerance and performance of the system when compared to a client-server variant.

While many agent systems use some way of encryption and authentication when agents are sent over the network, most of them lack a pubic key infrastructure (PKI). We address this issue in Sparta by providing such a PKI to manage our cryptosystem. Sparta utilizes an asymmetric (public/private key pair) cryptosystem to exchange private keys which are needed to secure agents when they are transfered over the network. The agent code is signed and can be authenticated before it is executed (to protect the platform). The signature

is also used to determine the set of permissions an agent is granted when executing on a platform.

In our opinion, the contribution of Sparta is the description of a system architecture to collect and correlate distributed data in an efficient way by using mobile agents. It can be used for intrusion detection, but other network applications are possible as well.

5 CONCLUSION

Although the possible advantages of mobile agents seem impressive at first, only a few systems use them to perform security related tasks. This stems from the fact that the benefits are not introduced automatically and often the disadvantages outweigh the intended improvements. IDA employs mobile agents mainly for tracing purposes while Micael and Sparta have more ambitious aims. In these systems, mobile agents actually carry out the event correlation.

Our goal in Sparta is to demonstrate that it is possible to beneficially apply agents to intrusion detection systems. We have recently finished a first prototype system to support our claims. The scalability, performance and fault tolerance can be improved when mobile agents perform distributed detection and don't need a central location where data is gathered. Nevertheless, the designer has to be careful that the amount of transfered data is not increased.

REFERENCES

[1] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Co., Box 42, Fort Washington, February 1980.

[2] M. Asaka, A. Taguchi, and S. Goto. The implementation of ida: An intrusion detection agent system. In *Proceedings of the 11th FIRST Conference*, June 1999.

[3] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. In *14th IEEE Computer Security Applications Conference*, December 1998.

[4] M. C. Bernardes and E. dos Santos Moreira. Implementation of an intrusion detection system based on mobile agents. In *International Symposium on Software Engineering for Parallel and Distributed Systems*, pages 158–164, 2000.

[5] J. D. de Queiroz, L. F. R. da Costa Carmo, and L. Pirmez. Micael: An autonomous mobile agent system to protect new generation networked applications. In *2nd Annual Workshop on Recent Advances in Intrusion Detection*, Rio de Janeiro, Brasil, September 1999.

[6] D. Denning. An intrusion-detection model. In *IEEE Symposium on Security and Privacy*, pages 118–131, Oakland, USA, 1986.

[7] W. Jansen and T. Karygiannis. Mobile agents and security. Special Publication 800-19, NIST, September 1999.

[8] W. Jansen, P. Mell, Karygiannis, and D. Marks. Applying mobile agents to intrusion detection and response. Interim Report (IR) 6416, NIST, October 1999.

[9] W. Jansen, P. Mell, T. Karygiannis, and D. Marks. Mobile agents in intrusion detection and response. In *12th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, June 2000.

[10] C. Krügel and T. Toth. A survey on intrusion detection systems. Technical Report TUV-1841-00-11, University of Technology, Vienna, 2000.

[11] C. Krügel and T. Toth. Sparta - a security policy reinforcement tool for large networks. In *submitted to I-NetSec 01*, 2001.

[12] W. Lugmayer. Gypsy: A component-based mobile agent system. In *8th Euromicro Workshop on Parallel and Distributed Processing (PDP 2000)*, Rhodos, Greece, January 2000.

[13] P. A. Porras and P. G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, October 1997.

[14] M. Roesch. Snort - lightweight intrusion detection for networks. In *USENIX Lisa 99*, 1999.

[15] G. B. White, E. A. Fisch, and U. W. Pooch. Cooperating security managers: A peer-based intrusion detection system. *IEEE Network*, pages 20–23, January/February 1996.