23

Analysis of JAVA Security and Hostile Applets

Dr. Klaus Brunnstein
Professor for Application of Informatics
University of Hamburg, Germany

Paper presented at SEC97 (Copenhagen, May 15, 1997)

Extended Abstract: **Rapid growth of Internet** was only possible when document description languages (esp. HTML), exchange protocols (HTTP) and **navigation tools** such as **Netscape`s browser and Internet Explorer** were available for mass usage. Basic Internet features (protocols, esp. TCP/IP, domain organisation and routing concepts), navigation tools and document description languages have been specified **without observing relevant security requirements**, esp. concerning **confidentiality** of sensitive processes and data. Moreover, essential safety aspects - **availability, reliability, maintainability, functionality** - have also been **neglected**. As security and safety are „design-inherent" features (i.e. they must be specified in design and enforced in implemented systems), later enhancements (such as IP v.6 including authentication and encryption, protocols such as S-HTTP, SSL or SET) can **at best reduce risks, but they can NOT cure past design faults**.

Within this insecure and unsafe Internet environment, **„agent" technologies** develop, which perform net-"work" with usually small processes which interoperate at an **assumed benefit of users**. A multitude of agents applications has been discussed, including delegation of tasks, handling email, coordination of group work and scheduling, mobile knowledge robots, distributed searches and many others. Early examples of agent technologies (though not named as such) have been XEROXs **worms** (which materialized in several network experiements and attacks) and **chain letters**. Started either automatically or from a users desktop (or better: WebTop), **agents work in hidden manners**. Therefore**, security and safety aspects** as well as **mechanisms to control agents** must be carefully analysed from design to implementation and actual work.

JAVA was announced in 1996, by Sun Microsystems (in a „White Paper") as 4G-language for Internet applications. It supports development and execution of small agents, called **„applets"** which are executed upon a specific software engine (conceptually similar to Niklaus Wirth's p-Code for Pascal).

According to Sun's summary:

> **Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded, and dynamic language."**

The C++-like JAVA-language has several deliberate **restrictions,** which according to Sun shall guarantee **applet security**. Among such **restrictions**, **access to files and Internet address space** (URLs) is strictly **prohibited**, and memory management (including garbage collection) is automatic; user-manipulated **pointers are not supported** in JAVA. As manipulation of memory (e.g. via memory residence) and manipulation of files are regarded as essential means with which (traditional) viruses propagate, some experts and Sun assume that **JAVA viruses are „impossible";** in a counterposition, others (such as Bill Cheswick) have regarded **JAVA as „ideal virus writing language".**

Besides language restrictions, JAVA offers **more security features**. A special class of services **„security.java"** supports encryption, authentication (digital signatures), secure key exchange and integrity mechanisms (checksumming). On this basis, applets may be authorized and authenticated. This provides a secure channel to the manufacturer which is „secure" if and when the manufacturer is regarded trustworthy. An additional feature is that JAVA applets are executed upon ist own interpretor; JAVA code can then be verified for conformance with security prescriptions (byte-code verifier).

With these enhancements, JAVA is **much better** than almost all other language systems though it is **inferior to Secure ADA** which offers also formal methods for proof of specified features (this is not foreseen in JAVA which does not hide its medium-level origin: it is similar to C/C++). Nevertheless**, JAVA applets are far from being „secure".** First, any **hidden manipulation** within the scope of the language is possible; it is therefore no surprise that **„hostile applets"** have soon been demonstrated on Internet, ranging from rather „innocent" Noisy.Bear" which „only" consumes processor time and memory, to „Killer-java" which installs multiple threads and kills some browsers. These applets can be classified as **„malicious trojan horses";** they do NOT replicate but may nevertheless harmfully affect user data and processes.

More generally, **„security" is a feature of a system whithin properly specified boundaries**. When JAVA applets **execute on insecure systems** (ranging from hardware to operationg systems to browsers and file systems), insecure use is possible despite JAVA restrictions. So far, **insecure interactions** of JAVA applets with browsers (esp. Netscape`s) have been discussed (Princeton University). So far, self-reproduction has not been demonstrated. Moreover, essential safety aspects - **availability, reliability, maintainability, functionality** - have also been **neglected in JAVA design.**

Conclusion: though JAVA has some security features, applets enlarges the risk of agent technologies. Based on insecure systems such as operating systems and browsers, risks of JAVA applets for sensitive information is significant.