



Vergleich: JAVA - ActiveX

Was is JAVA und ActiveX ?

- **Java** und **ActiveX** sind zwei Systeme, die es erlauben Computerprogramme mittels Web-Seite auszuführen.
- **Java** wurde von **JavaSoft** entwickelt (Division of **Sun Microsystems**).
- **ActiveX** wurde von **Microsoft** entwickelt.
- Zwischen **JavaSoft** und **Microsoft** gab es große öffentliche Diskussionen über Sicherheitsfragen der verschiedenen Systeme. Jeder der Hersteller beschuldigte den Anderen, dass er unsachgemäß mit Sicherheitsfragen umgehe. Diese Debatten haben zu einer Verunsicherung im IT-Markt geführt.

ActiveX Sicherheit

- ➔ *ActiveX bietet verschiedene Methoden, um die Ausführung von Programmen sicher zu gestalten. Normalerweise werden diese verschiedenen Varianten kombiniert um ein Maximum an Sicherheit und Flexibilität zu gewährleisten. Dies führt zu der Möglichkeit sehr granular verschiedene Sicherheitsstufen optimal an den Anforderungen moderner Unternehmen anzupassen.*
 - ➔ *Authenticode – ActiveX Programme besitzen eine digitale Signatur. Diese digitale Signatur vom Autor des Programms kontrolliert den Download des Codes auf eine Workstation. Somit können keine „fremden“ oder nicht autorisierte Programme geladen werden können.*
 - ➔ *“Administrator Approved” Einstellungen gibt jeder Sicherheitszone die Option nur Programme die von der IT-Abteilung zertifiziert wurden ausgeführt werden können.*
 - ➔ *CodeBaseSearchPath – Mit dem CodeBaseSearchPath Registry Key können, Administratoren kontrollieren wo das System gesperrt wird wenn ein **ActiveX download** geladen und ausgeführt wird.*
 - ➔ *IObjectSafety – The IObjectSafety interface bietet eine andere Methode neben “Safe for” flags“, um Operationen zu sichern die durch **ActiveX** ausgeführt werden können.*

ActiveX Sicherheit

- ➔ *“Safe for” flags* – Die “Safe for Initialization” und “Safe for Scripting” Flags, werden normalerweise von dem **ActiveX** control bei der Installation benutzt, um die Aktionen die eine Web-Seite ausführen kann zu definieren.
- ➔ *Kill bit* – Das “kill bit” ist ein Registry Wert, der verhindert, dass ein Browser ein **ActiveX** Control lädt. Es kann von keiner Security Zone Konfiguration überschrieben werden.
- ➔ *Security zones* – Die Security Zones Eigenschaft kann benutzt werden, um das Verhalten und die Adressierbarkeit der Controls zu definieren.
- ➔ *Windows Group Policy Objects (GPO)* – Group Policy Objekte können dazu verwendet werden, um Browser Konfigurationen zu definieren und Authenticode Settings zentral zu verwalten.

Java Sicherheit

- ➔ **Java** Sicherheit basiert komplett auf der Software Technologie. **Java** akzeptiert alle “downloaded” Programme und führt diese in einer Sicherheits „Sandbox“ aus. Eine “Sandbox” ist ein Sicherheitszaun, der das Programm umgibt und es davon hindert auf private Daten zuzugreifen. Solange kein Loch im Zaun ist, ist man mit dieser Methode sicher.
- ➔ **Java** Sicherheit basiert auf der korrekten Implementierung der Software. Die Software sichert die korrekte Funktion der Sandbox.
- ➔ **Java**-enabled Produkte beginnen digitale Signaturen zu benutzen. Die Idee ist wie bei **ActiveX**: Programme sind digital signiert und man kann auf Basis dieser Signaturen entscheiden ob man einem Programm mehr Rechte erteilt.

Nachteile - JAVA

Java arbeitet mit einer „Virtual Machine“ : diese zusätzliche Softwareebene hat folgende Auswirkungen:

- ➔ **JAVA** ist kompiliert und dies „Just in time (JIT)“ ; d.h. um Java auszuführen werden mehr Ressourcen benötigt und dies kann zu einer Verlangsamung der Computer führen; speziell wenn die Computer über wenig Ressourcen verfügen. Durch den höheren Speicherbedarf kann es zu erhöhten “Paging” kommen und damit alle Anwendungen stark verlangsamen.
- ➔ **JAVA** kann die verschiedenen System Ressourcen nicht direkt ansprechen, da Java nicht in der Lage ist die tieferen System Layers zu adressieren. Die resultierenden „translation“ Operationen beeinträchtigen ebenfalls die Systemperformance und können in schlimmsten Fall zu Systeminstabilitäten führen, wenn Hardware angesprochen werden muss.
- ➔ **JAVA** applets benötigen die **Java** Runtime Environment (JRE), um auf allen Clients installiert werden zu können.
- ➔ Die JRE benötigt regelmäßige Updates und Bugfixes, um die Sicherheit Aufrecht zu halten. Hierdurch kann es zu Versionskonflikten kann da eine Anwendung der Version A die andere Anwendung jedoch noch Version B benötigt.

Nachteile ActiveX

- ➔ **ActiveX** controls laufen nicht in einer „virtual machine“ und können daher zumindest Theoretisch mehr Schaden verursachen als Java applets. Dies kann jedoch nur geschehen, wenn es keine Sicherheitsrichtlinien gibt, wie dies oft in Privatumgebungen zu finden ist.
- ➔ Portability: **ActiveX** läuft primär nur auf Windows Systemen und ist unterstützt zurzeit noch nicht Mac OS oder Linux.
- ➔ Wenn Sicherheits-Bugs oder Löcher gefunden werden benötigt man im Allgemeinen ein Systemupdate. Diese Updates sind meist Part eines Servicepacks des Betriebssystems..

Vergleich – allgemeine Funktionen

- **Java** ist einfacher zu portieren und unterstützt mehr Betriebssysteme wie Linux und Mac OS
- Beid **Java** and **ActiveX** benötigen regelmäßige Sicherheitsupdates. **ActiveX** Sicherheitsupdates sind im Allgemeinen Teil des Servicespacks des Betriebssystems, die ohnehin benötigt werden. **Java** patches werden separat installiert.
- **Java** Versionen sind öfter nicht kompatibel – entweder untereinander oder mit verschiedenen Anwendungen.
- Im Gegensatz zu **Java** ist **ActiveX** precompiled und benötigt weniger Systemressourcen.
- **ActiveX** kann Hardware Abstrationlayer direkt ansprechen und benötigt keine „translation“ wie **Java**, was der Systemstabilität zu Gute kommt.

Vergleich – Sicherheit

- ➔ Signierter **Java** Code ist noch nicht weit verbreitet und besonders akzeptiert in der IT Community.
- ➔ **Microsoft's** Sicherheitsmodel funktioniert nur wenn es sinnvolle Sicherheitsrichtlinien über Policies eingehalten werden. Daher ist dies ideal für Corporate Environments. Bei Privatanwendern müssen diese selbst entscheiden welchen Anwendungen ausgeführt werden dürfen. Dies stellt ein Sicherheitsrisiko dar.
- ➔ Auf Grund der Tatsache, dass **Java** sehr portable ist, erschwert es speziell in Unternehmen die Kontrolle über zentralisierte Policies sehr. Nur auf eine Software „Sandbox“ zu vertrauen ist ein hohes Risiko für IT Abteilungen.
- ➔ Da das **Java** Sicherheitsmodel Software basierend ist, kann es nur so gut sein wie eine Software selbst. Die Hauptgefahr liegt in der Komplexität der Implementierung der Sandbox. Im Allgemeinen heißt es, je komplexer eine Technologie, desto höher die Wahrscheinlichkeit von Fehlern.

Zusammenfassung

- **ActiveX** und **Java** sind beide gleich unsicher im Bezug auf Sicherheitslöcher
- **ActiveX** ist sicher in Firmen Umgebungen. Die Verwendung von strengen Sicherheitsrichtlinien (Policies) und “Code signing” machen Angriffe theoretisch unmöglich. **Java** ist schwierig in Firmenumgebungen zu kontrollieren.
- **Java** ist sicher in Privat Umgebungen durch die „Sandbox“ Architektur
- **ActiveX** hat eine bessere Performance und Stabilität wenn es darauf ankommt Hardware anzusprechen.
- **Java** ist portabler – dies bedeutet aber auch, dass häufiger Patches eingespielt werden müssen und das die Möglichkeit zu Inkompatibilitäten gibt.
- **ActiveX** kann zentral von Group Policies kontrolliert werden. Änderungen können zentral ausgeführt werden und Änderungen an Sicherheitsrichtlinien greifen sofort..