Aktive Webinhalte und Schutzmöglichkeiten

Konstantin Agouros
elwood@agouros.de

21. Mai 2000

1 Einleitung

Mit dem Siegeszug des World Wide Web wurde die Nachfrage nach größerer Interaktivität immer stärker. So wurden zunächst Webserver um den CGI-Mechanismus erweitert (und HTML um die FORMS-Elemente), und es wurden Browser-Plugins eingeführt, die über den MIME-Mechanismus wie bei Emails externe Applikationen starten. Schließlich wurden die Browser um ausführbare Inhalte erweitert, und mit Javascript wurde die erste Programmiersprache in HTML und in einen Webbrowser integriert. Dem folgten Java von Sun und ActiveX sowie VBScript von Microsoft.

In diesem Beitrag werden zunächst die verschiedenen aktiven Inhalte vorgestellt. Dabei wird betrachtet, wie die verschiedenen Inhalte im HTML-Kontext auftreten und in welchem Ablauf-Kontext sie ablaufen. Es werden das Sicherheitsmodell, Sicherheitslücken und bei den Programmiersprachen der Sprachumfang betrachtet. Dabei wird auch Augenmerk auf die Zusammenhänge zwischen den verschiedenen Technologien geworfen.

Es wird untersucht, welche Schutzmöglichkeiten es gibt, wobei zentraler Schutz und individueller Schutz dargestellt werden. Hier werden auch die Machbarkeit, bzw. die Probleme eines solchen Schutzes untersucht, sowie ein Blick auf Performanceverluste, die Komplexität des Problems sowie Anforderungen aus der Praxis gestellt.

Nach einem Blick auf die "Psychologische Komponente" (wie und wann wird der Benutzer auf die Ausführung aktiver Inhalte aufmerksam gemacht, bzw. kann man dies verhindern) werden in einer Übersicht eine Auswahl bestehender Filterprodukte vorgestellt und hinsichtlich ihrer Filtermöglichkeiten, der Effizienz und der Bedienbarkeit verglichen. Zum Abschluß werden aus diesen Betrachtungen davor einige Forderungen abgeleitet.

2 Aktive Inhalte

In diesem Abschnitt soll der Begriff "Aktiver Webinhalt" definiert und abgegrenzt werden.

2.1 Abgrenzung

Unter einem aktiven Webinhalt soll im folgenden ausführbarer Code verstanden werden, der von einem Webbrowser heruntergeladen wird und im Programmkontext dieses Webbrowsers ausgeführt wird. Hierbei sind keine Inhalte gemeint, die ein externes Programm zur Ausführung benötigen (z.B. ein Word-Dokument mit Makrodefinitionen, obwohl solche Inhalte auch ein erhebliches Gefahrenpotential bilden).

Dieser Beitrag untersucht auch nicht die Problematik von ausführbaren Programmen, die zunächst heruntergeladen und dann ausgeführt werden [1]. Ebenfalls ausgeklammert ist die Virenproblematik, da das Internet nur eine neue Möglichkeit neben Disketten und CDs darstellt, Viren auf den Computer zu bekommen.

2.2 Risikofaktoren

Das unkontrollierte Ausführen von Programmen stellt natürlich ein Risiko dar. Dies ist auch den Entwicklern der Browser bewußt gewesen (oder gemacht worden), so daß Sicherheitsmechanismen implementiert wurden, die den Funktionsumfang auf "ungefährliche" Funktionalitäten einschränken oder vor potentiell gefährliche Aktionen oder Funktionsbereiche (z.B. Dateisystemzugriffe) Sicherheitsabfragen stellen.

Dabei treten jedoch folgende Probleme auf:

- Konzeptionelle Fehler im Entwurf der Sicherheitsmechanismen
- Unerfahrenheit bzw. Unachtsamkeit der Benutzer bei der Konfiguration der Sicherheitskomponenten und der Beantwortung von Sicherheitsabfragen.

Für den Systemadministrator eines (durch eine Firewall) an das Internet angeschlossenen LANs ergeben sich so folgende Möglichkeiten, Systeme am LAN vor böswilligem Code zu schützen:

1. Jeder einzelne Arbeitsplatz wird entsprechend konfiguriert und abgesichert, und die Benutzer werden so geschult, daß sie verantwortungsvoll mit aktiven Inhalten aus dem Internet umgehen können.
2. Die Filterung der von außen kommenden aktiven Inhalte geschieht an einer zentralen Stelle auf oder neben der Firewall. Dabei werden sämtliche Inhalte abgeblockt.
3. Es wird ein eigenes Netz von "Surf-Stationen" eingerichtet, die vom LAN physisch getrennt sind und in denen böswilliger Code keinen Schaden anrichten für das Produktnetz kann.

Die erste Variante wäre zwar wünschenswert, ist aber aus praktischen Gründen nicht realisierbar. Die zweite Variante ist in der Praxis nur bedingt durchsetzbar, da viele "wichtige" Webseiten nur noch mit aktiven Inhalten benutzbar sind (eine rühmliche Ausnahme stellen hier z.B. die Websites von Sun und IBM dar, die lange in dreifacher Ausfertigung existierte: Normal mit Bildern, nur Text und eine Java-Version). Die dritte Möglichkeit schließlich ist recht kostspielig, läßt aber uneingeschränkten Zugriff auf das Internet ohne Gefahren für ein LAN zu. Dabei sollten natürlich Dinge wie Virenschutz weiterhin berücksichtigt werden, um zu verhinden, daß aus diesem ungefilterten Netz heruntergeladene Daten über den Umweg Diskette nicht doch noch Schaden anrichten.

3 Die Technologien

In diesem Abschnitt werden die Sprachen bzw. die Technologien vorgestellt. Dabei werden zunächst Java und ActiveX und schließlich die Scriptsprachen in der Reihenfolge der Komplexität der Filterung behandelt.

[1] wenngleich die Implementierung des Downloads im MS Internet Explorer und dem daraus möglichen Start eines externen, heruntergeladenen Programms der diskutierten Problematik sehr ähnlich ist

3.1 Java

Java Virtual Machines (JVMs) in Webbrowsern traten zum ersten mal in der Version 2 des Netscape Navigators auf [2]. Heute stehen JVMs für Browser aller gängigen Betriebssysteme zur Verfügung. Das folgende Beispiel zeigt HTML-Code, der zum Start der JVM führt, die erste Javaklasse herunterlädt und sie startet.

```
<HTML>
<BODY>
<APPLET CODE=DEMO.CLASS>
< /APPLET>
< /BODY>
< /HTML>
```

Die folgende Abbildung soll den Kommunikationsverlauf, der zur Ausführung einer Javaklasse führt, verdeutlichen. Die erste Klasse enthält eine Referenz auf die Klasse demo2.

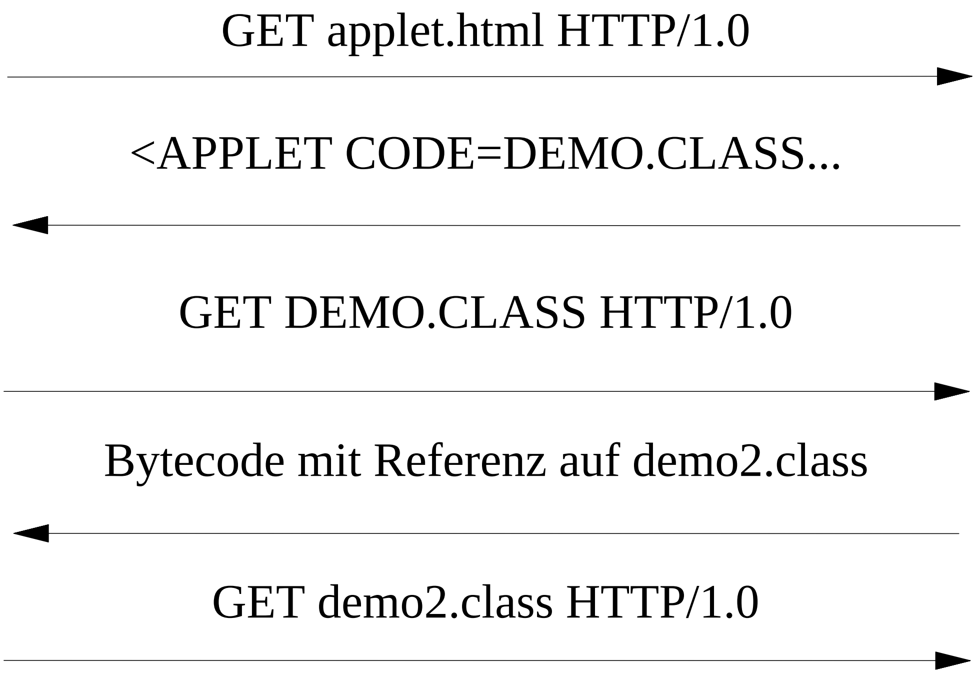

Abbildung 1: Kommunikationsablauf bei Java

Der Applet-Tag kann verschiedene Parameter enthalten. Entscheidend ist der *CODE* Parameter bzw. der *Archive* Parameter, der ein JAR-Archiv, eine Datei im ZIP-Format, die eine oder mehrere Klassendateien sowie eine digitale Signatur im X.509-Format enthalten kann. Die Programmiersprache Java enthält den vollen Funktionsumfang einer modernen Programmiersprache. Dies beinhaltet auch Schnittstellen, um auf Dateien zuzugreifen oder Netzwerkverbindungen zu öffnnen. Dem stellt Java jedoch ein Sicherheitsmodell gegenüber:

[2] Der erste javafähige Webbrowser war natürlich Suns HotJava

- Javaklassen, die als "nicht vertrauenswürdig" eingestuft werden, laufen in einem eigenen Speicherbereich ab, von dem aus sie keinen Zugriff auf außerhalb liegende Speicherbereiche haben.
- Standardklassen können nicht von von außen kommenden Klassen überschrieben werden, um so zu verhindern, daß trojanische Pferde zum Ablauf kommen.
- Wenn eine Javaklasse etwas Sicherheitskritisches ausführen möchte, wie eine Netzwerkverbindung aufbauen oder eine Datei öffnen, muß dies erst vom Securitymanager, einer Komponente der JVM, freigegeben werden.
- Die Entscheidung über die Ausführungsberechtigung wird entweder aufgrund der Quelle des Applets getroffen (über das Netz heruntergeladene Applets haben keinen Dateizugriff und dürfen Netzwerkverbindungen nur zu dem Server aufbauen, von dem sie heruntergeladen wurden)
- Dabei können je nach Browser diese Unterscheidungen nach lokalen und entfernten (hinter einer Firewall liegenden) Servern als Quelle getroffen werden.
- Eine Konfigurationsmöglichkeit, in der eine Matrix konfiguriert werden kann (Dateizugriffe sollen von *.sun.com erlaubt werden, Netzwerkzugriffe bei Applets von *.dfn.de), gibt es nicht.
- In neueren Browsern gibt es in der Netscape-Klassenhierarchie die Möglichkeit, einen sogenannten "Grant" anzufordern, der den Securitymanager dazu veranlaßt, diese Aktion zuzulassen, obwohl es eigentlich gegen die Securitypolicy verstößt. Der Benutzer kann dann in einer Alertbox entscheiden, ob die Aktion erlaubt werden soll oder nicht.

Hierbei treten einige Probleme auf:

- Ist der Securitymanager fehlerhaft implementiert, so kann er umgangen werden.
- Unklare Warnmeldungen können leicht dazu führen, daß unerfahrene Benutzer eine böswillige Aktion genehmigen.
- Denial of Service Attacken durch den Verbrauch von Systemressourcen wie Speicher, CPU-Zeit, Threads, Prozesse und Fenster sind möglich. Es gibt zwar erste Ansätze, hier zu limitieren, wieviel Instruktionen ein Applet ohne Unterbrechung ausführen kann. Diese Einschränkungen können aber leicht umgangen werden.
- Mit der DirectJ Klassenhierarchie von Microsoft sind bei den Windows Betriebssystemen Zugriffe direkt auf Betriebssystemebene möglich.
- Es gibt Methoden, die andere Webseiten in den Webbrowser laden. Diese Methoden sind nicht vom Securitymanager betroffen. Hiermit können z.B. kostenpflichtige Seiten, für die einmal ein Paßwort eingegeben wurde, immer wieder aufgerufen werden.
- Es gibt nicht abgesicherte Funktionen, die zur Datenübertragung nach außen benutzt werden können:
  - Das Öffnen von URLs a'la: `http://www.boeserserver.de/cgi-bin/sammelscript.cgi?der+server+hier+heisst+hansi`
  - Hostnamelookups a'la: `der-server-hier-heisst-hansi.boesedomain.de`

  Letztere Methode scheint auf den ersten Blick unauffälliger, aber das Java-Applet kann den String auch verschlüsseln, so daß die Datenübertragung in den Logfiles des Webproxies nicht so auffällig ist.

3.1.1 Zusammenhänge mit anderen Technologien

Unterhalb der Netscapeklassenhierarchie gibt es die Möglichkeit, Javascriptcode auszuführen. Die Klasse `netscape.javascript.JSObject` enthält eine `eval` Methode, die als Argument einen String mit Javascriptcode übergeben bekommt, der dann ausgeführt wird.

Die Konsequenz hieraus ist, daß es wenig Sinn macht, Java in einem Filter zu erlauben, aber JavaScript zu verbieten. In den individuellen Einstellungen des Browsers ist dies allerdings durchaus sinnvoll.

3.1.2 Datenübertragung und Filterung

Die offensichtlichste Methode ist natürlich das Herausschneiden des APPLET-Tags. Wie im Javascriptabschnitt aber noch deutlich wird, ist dies unzureichend, wenn differenziert zwischen Java und Javascript gefiltert werden soll. Deswegen ist es auch nötig, eine Möglichkeit zu besitzen, Javabytecode zu erkennen und abzublocken.

Einzelne Javaklassen werden als Datei mit dem MIME-Type "application/octet-stream" und der Dateiendung ".class" übertragen. Dieser MIME-Type wird allgemein für Binärdateien unbekannten Typs verwandt und ist deshalb als Filterkriterium nicht sehr gut verwendbar [3]. Angreifer können außerdem bei einer Man-in-the-middle-Attack diesen Type modifizieren. Die Dateiendung kann verwandt werden, da eine JVM, die kein JAR-Archiv lädt, versucht, bei der Instanziierung einer neuen, noch nicht geladenen Klasse eine Datei mit dem Namen der Klasse und der Endung ".class" herunterzuladen. Dies ist daher als Filterkriterium geeignet, da so das Filterprogramm bereits verhindern kann, daß die unerwünschte Klasse überhaupt vom Webserver geladen wird.

Schließlich bleibt noch das Erkennen des Java-Bytecodes an Hand seines Dateiformates. Die ersten vier Byte einer Klassendatei müssen die "Magic Number" 0xCAFEBABE sein. Dem folgen je zwei Bytes für die Major und die Minor Release der JVM, mit der die Dateien erzeugt wurden, und für die sie geschrieben wurden. Aufgrund von lediglich 8 Byte kann also hier recht effizient eine Entscheidung getroffen werden, die sogar noch eine Versionskontrolle enthält (vielleicht ist ja Java 2.5 fehlerfreier als Java 2.2).

Seit dem JDK 1.1 gibt es JAR-Dateien. JAR-Dateien sind Archivdateien, die mehrere Javaklassen enthalten. Zusätzlich kann eine X.509-Signatur enthalten sein, die den Autor des Codes identifiziert. Dieses Zertifikat wird bei der Anforderung eines Grants zur Freischaltung von Funktionalitäten mit angegeben. In JAR-Archiven kann allerdings auch Javascript-Code enthalten sein, so daß ein generelles Abblocken eine Differenzierung der beiden Technologien ausschließt.

Die Signatur bietet ein weiteres Filterkriterium, welches recht interessant ist, da hier über die Freigabe einer als vertrauenswürdig eingestuften Certificate Authority relativ effizient gearbeitet werden kann. Ist die CA, die die Identität eines Programmautoren unterschrieben hat, jedoch nicht vertrauenswürdig, so nützt auch dieses nichts.

Problematisch hierbei ist jedoch eine erhöhte Komplexität und Verarbeitungszeit im Filter, da jede JAR-Datei zunächst ausgepackt werden muß, um eine Entscheidung zu treffen. Soll auch noch die Signatur als Kriterium herangezogen werden, so erhöht sich der Aufwand noch einmal nicht unerheblich, da die Verifikation einer X.509-Signatur, verglichen mit dem Vergleich von 8 Byte, recht aufwendig ist.

Da das Bytecodeformat standadisiert ist, ist es auch möglich, durch Analyse herauszufinden, welche Klassen ein Applet benutzt. Dies kann als Filterkriterium verwandt werden, um z.B. Applets abzublocken, die Datei- oder Netzwerkklassen benutzen wollen, oder auch, ganze Klassenhierarchien (com.ms.*) zu sperren.

[3] neuerdings gibt es auch den MIME-Type "application/java-vm"

3.2 ActiveX

Microsofts Technologie, die mit dem Internet Explorer 3.0 eingeführt wurde, muß etwas differenzierter betrachtet werden. Ähnlich wie bei Java wird im HTML-Code durch einen Tag das Herunterladen und der Start des ausführbaren Codes, im Microsoft-Sprachgebrauch ein "Control", angestoßen. ActiveX steht im Microsoft Internet Explorer für die Microsoftbetriebssysteme, MacOS und einige Unix-Derivate zur Verfügung. Außerdem gibt es für den Netscape Browser auf den Microsoftplattformen Plugins. Im folgenden ein HTML-Beispiel für ActiveX:

```
<HTML>
<BODY>
<OBJECT CLASSID=''xyz.abc''
CODE=Dateiname
>
< /BODY>
< /HTML>
```

Der eigentliche aktive Inhalt kann dabei eine Binärdatei für die Zielplattform, ein J- oder VBScript-Programm, oder aber Javabytecode sein. Javabytecode wird dabei der JVM des Browsers übergeben, die Scriptsprachen der entsprechenden Scriptingengine. Bei Binärdateien wird aufgrund des Classid-Parameters in der Registry geprüft, ob dieses Control schon installiert ist. Ist dies der Fall, so wird es gestartet, andernfalls wird es nach Rückfrage beim Benutzer heruntergeladen und installiert[4]. Im Falle der 3.0-Version des Browsers wurde ohne Rückfrage beim Benutzer heruntergeladen und installiert.

Eine Authentisierung des Codes bzw. eine Identifizierung des Autors kann mittels Microsofts Authenticode-Technologie ähnlich wie bei den JAR-Archiven realisiert werden. In eine Cabinet (CAB) Datei wird eine digitale Signatur eingefügt, die dem Benutzer vor der Installation bzw. Ausführung bei Javabytecode präsentiert wird. Die Einschränkungen, die für über den APPLET-Mechanismus heruntergeladene Javaprogramme gelten, sind auch hier bei Bytecode wirksam, allerdings mit einer Einschränkung: Im Netscape-Sicherheitsmodell muß jede einzelne Aktion durch Beantragen eines Grants und Bestätigung des Benutzers freigegeben werden. Im ActiveX-Modell akzeptiert der Benutzer einmalig, daß dem Code vertraut werden kann, danach gelten keine Einschränkungen mehr.

3.2.1 Risiken

Im Falle eines Javacontrols gelten die gleichen Bedenken wie bei reinem Java, wenngleich das Risiko durch besseren Support von betriebssystemnahen Klassen (DirectJ) und dem Sicherheitsmodell, das nicht einzelne Aktionen, sondern ein ganzes Archiv bzw. Programm freischaltet, erhöht ist. Im Falle eines Binärprogramms gibt es überhaupt keine Restriktionen mehr. Es können Dateien auf dem lokalen Rechner manipuliert oder gelesen werden oder Rechner im lokalen Netzwerk können systematisch angegeriffen werden. Durch direkte Integration der DCOM-Technologie ist sogar Zugriff auf lokal laufende Applikationen möglich. Bei Betriebssystemen mit mangelnden Schutzmechanismen (z.B. Windows 95) können auch privilegiert Aktionen ausgeführt werden (z.B. Verbindungsannahme auf Port < 1024), was das Ausspähen von Daten, die über Broadcastpakete versandt werden, ermöglicht. Diese können dann auch ohne größere Probleme mit den bereits vorgestellten Mechanismen oder direkt per email oder ftp versandt werden. Auch die Installation von Viren oder ausgefeilteren Spionageprogrammen ist möglich. Technologien wie Active Desktop ermöglichen es böswilligen Controls abzulaufen, selbst, wenn der Benutzer den Browser gar nicht bewußt gestartet hat.

Dr. Kai Fuhrberg vom BSI schreibt in [5]: "Zusätzlich ist für die meisten Hersteller von Software (und hierzu gehören auch Betriebssysteme!) Sicherheit nach wie vor nur ein Randthema. Daher ist die

[4] Ob dies stattfindet, hängt von den Sicherheitseinstellungen des Browsers ab

Unterstützung bei der Absicherung des eigenen Rechners oder eines Firmennetzes sehr unzureichend. Als Beispiel muß hier speziell die Firma Microsoft genannt werden, die durch neue 'Technologien' wie Makroviren und ActiveX gezeigt hat, daß ab einer gewissen Marktbeherrschung keine Rücksichten auf 'Randthemen' wie Sicherheit notwendig sind.

Ein ActiveX-Control, das recht grafisch deutlich macht, was alles möglich ist, befindet sich unter http://housecall.antivirus.com. Dies ist ein Virenscanner, mit dem man sogar alle Netzwerklaufwerke untersuchen kann. Dieses Control ist zwar gutmütig und nützlich, zeigt aber in erschreckender Weise, daß vor dieser Technologie nicht sehr viel sicher ist.

3.2.2 Zusammenhänge mit anderen Technologien

Durch die Vielfalt der Sprachen, in denen ein Control geschrieben sein kann, treten hier wesentlich heftigere Verwicklungen auf. Neben den bereits im Java-Abschnitt beschriebenen Möglichkeiten, evtl. vorhandene Filtersperren zu umgehen, gibt es bei ActiveX sogar die Möglichkeit eigene Scriptingengines einzuklinken, so daß schließlich in einer Webseite gleich Perlcode stehen könnte, der dann vom Browser nach Herunterladen des entsprechenden Controls ausgeführt wird.

3.2.3 Datenübertragung und Filterung

Auch hier ist das einfachste, den <OBJECT>-Tag zu filtern. Wie bei Java ist es allerdings auch hier wünschenswert, die Binärinhalte zu filtern, da sonst durch dynamische HTML-Generierung der Filter unterlaufen werden kann. Der ausführbare Code wird dabei als Binär- oder ASCII-Datei übertragen. Dieses uneinheitliche Datenformat macht ein effizientes Abblocken praktisch unmöglich.

Auch Dateiendungen können nicht verwandt werden, da diese genauso uneinheitlich sind. Bei der Inhaltsanalyse bleibt im Binärfall lediglich die digitale Signatur. Im Javafall greifen die im vorherigen Abschnitt beschriebenen Technologien.

3.3 JavaScript

JavaScript ist eine einfache objektorientierte Scriptsprache, die zuerst im Netscape Navigator (ab Version 2) und später auch im Internet Explorer (ab Version 3) implementiert wurde. Dabei gibt es zwischen den Browsern und den Versionen der Browser (bzw. der Versionen von JavaScript) Inkompatibilitäten in der Syntax und zum Teil auch in der Semantik der Sprache. Es gibt für JavaScript einen ECMA-Standard. Allerdings ist die Browserentwicklung zum Teil schon wieder weiter als der Standard. Die Sprache umfaßt Standardkontrollelemente (if, while, ...), eine Objektorientierung in Form von Hashes, reguläre Ausdrücke im Funktionsumfang wie Perl, Funktionen zum dynamischen Erzeugen und Darstellen von HTML, dem Öffnen von neuen Browserfenstern und dem Öffnen von URLs. Außerdem können beim Netscape Communicator nach Anforderung eines Grants (wie bei Java) verschiedene Daten des Browsers ausgelesen und auch verändert werden. Trifft der Browser auf JavaScriptcode in einer der möglichen Ablaufumgebungen, so wird dieser ausgeführt.

JavaScript ist in der Art des Auftretens wohl am vielseitigsten. Es gibt insgesamt 6 verschiedene (dokumentierte) Möglichkeiten, an denen JavaScript auftreten und zur Ausführung kommen kann. Diese Möglichkeiten sind:

Code innerhalb eines SCRIPT-Tags Eingeschlossen von einem <SCRIPT>-< /SCRIPT> Paar steht der auszuführende Code. In älteren Seiten ist er noch von HTML-Kommentaren umgeben (<!−−−− >), damit Browser, die den Tag nicht kennen, nicht darüber stolpern. Alternativ kann der SCRIPT-Tag ein SRC oder ARCHIVE-Attribut enthalten, mit dem eine JavaScript-Datei oder ein JAR-Archiv nachgeladen werden können. Wenn eine Datei heruntergeladen wird, so wird sie ausgeführt, als ob der Code innerhalb des Tags gestanden hätte. Das JAR-Archiv kann auch hier eine X.509-Signatur enthalten, die bei der Freigabe von Grants zur Authentifizierung benutzt werden kann.

Eventhandler Die zweite, häufig verwandte Möglichkeit sind sogenannte "Eventhandler". Sie sind optionale HTML-Attribute. Sie haben immer ein Argument. Der Name des Attributs gibt das Ereignis an, das Argument ist der ausführbare Code. Häufig ist dies nur ein Funktionsaufruf einer in einem SCRIPT-Bereich definierten Funktion. Jedoch können auch durch Semikolons getrennte Befehle dort stehen. Die standardisierten Eventhandler sind: `onAbort`, `onBlur`, `onChange`, `onClick`, `onDblClick`, `onDragDrop`, `onError`, `onFocus`, `onHelp`, `onKeyDown`, `onKeyPress`, `onKeyUp`, `onLoad`, `onMouseDown`, `onMouseOut`, `onMouseOver`, `onMouseUp`, `onMove`, `onReset`, `onResize`, `onSubmit`, `onUnload`.

Microsoft Internet Explorer versteht (vermutlich unter anderen) auch noch folgende: `onCellChange`, `onDataAvailable`, `onDatasetChanged`, `onDataSetComplete`, `onDrag`, `onDragEnd`, `onDragEnter`, `onDragLeave`, `onDragOver`, `onDragStart`, `onDrop`, `onLoseCapture`, `onPropertyChange`, `onReadyStateChange`, `onRowEnter`, `onRowExit`, `onRowsDelete`, `onRowsInserted`, `onScroll`, `onSelectStart`.

JavaScript-URLs Eine seltener benutzte Methode sind URLs, die als Protokollanteil "javascript:" gefolgt vom Code enthalten. Diese URLs können an jeder Stelle stehen, an der Verweise auf andere Dateien oder URLs enthalten sind. Der Code wird bei <A> Tags erst ausgeführt, wenn der Link angeklickt wird.

Entities HTML-Entities dienen zur Darstellung von Sonderzeichen in einer Webseite. Dazu gehören z.B. deutsche Umlaute oder auch die spitzen Klammern selbst. Sie beginnen mit der Zeichenkombination `&{`. JavaScript-Entities sind eigentlich dafür gedacht, den Wert eines Argumentes für ein HTML-Attribut dynamisch zu generieren. Beispiel:

```
<BODY BGCOLOR=''&{Code zum generieren der Farbe}''>
```

Der geöffneten Klammer muß dabei eine geschlossene folgen. Dabei muß dieser Teil in Anführungszeichen stehen. Auf so eingepackten Code kann man auch in sogenannten "Conditional Comments" stoßen. Dies sind HTML-Kommentare, denen ein `&{...}` Block folgt, der mit einer Abfrage enden muß.

Cascading Stylesheets Cascading Stylesheets (CSS) wurden mit HTML 3.2 eingeführt, um einheitliches Webdesign zu vereinfachen. Mit ihnen ist es z.B. möglich, zu definieren, daß für alle Listen die Schriftfarbe immer blau ist. Wird ein solcher Style definiert (mittels des <STYLE> Tags), so kann der Tag auch ein Attribut `TYPE=text/javascript` enthalten und darauf folgt JavaScript-Code, ähnlich wie beim Style-Tag, der ausgeführt wird. Diese und die folgende Technologie sind auch unter dem Stichwort JavaScript Style Sheets (JSS) bekannt.

LINKS Ähnlich wie bei CSS kann auch hier ein Attribut `TYPE=text/javascript` stehen, dem ein `CODE=` Attribut folgt. Hier gelten die gleichen Probleme wie bei Entities.

Der folgende HTML-Code illustriert noch einmal alle diese Methoden:

```
<!DOCTYPE HTML PUBLIC "-//AdvaSoft//DTD HTML 3.2 extended 961018//EN">
<HTML>
<HEAD>
 <TITLE>Ampersand-TEst</TITLE><SCRIPT>alert("JavaScript geht als script");</SCRIPT>
 <STYLE TYPE="text/javascript">
 alert("JavaScript geht als Style")
```

```
 </STYLE>
<link href='alert.js' rel=stylesheet type=text/javascript>
</HEAD>

<BODY BACKGROUND="&{alert("JavaScript geht als Entity");};">
<!--&{alert("JavaScript geht als Conditional Comment");1==1};
Test
-->

<P>

<A HREF="javascript:alert('JavaScript geht als Link')">LinkTest</A>
<FORM>
<INPUT TYPE=text onChange="alert('JavaScript geht als EventHandler')">
</FORM>

</BODY>
</HTML>
```

3.3.1 Risiken

Abgesehen von möglichen Fehlern in Browsern treten folgende konzeptionelle Risiken auf:

1. Denial of Service Attacken durch Resourcenverbrauch. Hierbei gibt es zwar Limitierungen der Browser bezüglich der Anzahl hintereinander ausführbarer Kommandos, aber diese lassen sich durch geschickte Programmierung leicht umgehen.

2. Das unerwünschte Öffnen anderer Webseiten mit evtl. kostenpflichtigem Inhalt. Hat ein Browser etwa Word als Applikation für den MIME-Type "application/ms-word" eingetragen, so kann man auf diesem Weg den Start der Applikation Word provozieren und so sogar das Herunterladen unerwünschter Dokumente veranlassen.

3. Umgehung von evtl. bestehenden Filtern für bestimmte Inhalte an einer Firewall, indem dynamische Inhalte generiert werden. Siehe folgenden Code:

   ```
   <!DOCTYPE HTML PUBLIC "-//AdvaSoft//DTD HTML 3.2 extended 961018//EN">
   <HTML>
   <HEAD>
    <TITLE>Java-Test</TITLE>
   <SCRIPT>
   document.write("<");
   document.write("A");
   document.write("PPLET\n");
   document.write("name=Webreport\ncode=Webreport.class\n
   codebase=http://server/webaccess/\nwidth=800\n
   height=500\nalign=Top>");
   document.write("</APPLET>");
   </SCRIPT>
   </HEAD>
   <BODY>
   </BODY>
   </HTML>
   ```

Dieser Code lädt und startet das Applet Webreport, nachdem dynamisch der APPLET-Tag in die aktuelle Webseite generiert wurde.

4. Manipulation der Status-Zeile des Browsers, was dem Benutzer das Ziel eines Links verschleiert.
5. Veränderung der Browsereinstellungen (*)
6. Datei-Diebstahl durch Manipulation von File-Upload-Fields (*)
7. Zugriff auf die History-Einstellungen des Browsers und die Inhalte anderer Browserfenster mit unter Umständen vertraulichem Inhalt (*)
8. Versandt von Emails (*)

Die mit einem * gekennzeichneten Punkte sind nur im privilegierten Modus, also nach Erteilung eines Grants, möglich. An diesen ist eine digitale Signatur gekoppelt, deren Informationen bei der Erteilung eines Grants dem Benutzer präsentiert werden.

3.4 Zusammenhänge mit anderen Technologien

Hier gibt es zwei zu betrachtende Möglichkeiten:

1. Java-Aufrufe direkt aus JavaScript heraus
2. Dynamische HTML-Erzeugung zum Start von Java oder ActiveX, um einen TAG-Filter zu unterlaufen

Die erste Möglichkeit wird etwa an folgendem Beispiel deutlich:

```
var f = new java.awt.Frame(''Hello World'');
var ta = java.awt.TextArea(''hello, world'',5,20);
f.add(''Center'', ta);
f.pack();
f.show();
```

In dieser Art und Weise werden auch Grants für JavaScript beantragt, indem die entsprechenden Funktionen unterhalb der Netscape Klassenhierarchien aufgerufen werden. Hieraus folgt, daß man keinen Schutz vor Java hat, wenn man an einem Filter Java aber nicht JavaScript herausfiltert. Hier ein Codebeispiel aus [3], welches ziemlich deutlich zeigt, welches Potential darin steckt:

```
<HTML><BODY><SCRIPT>
funcition save_it(filename, filedata);
{
netscape.security.PrivilegeManager.enablePrivilege(''UniversalFileWrite'');
var jo_file = new java.io.FileOutputStream(filename);
var jo_stream = enw java.io.DataOutputStream(jo_file);
jo_stream.writeChars(filedata);
jo_stream.close();
netscape.security.PrivilegeManager.disablePrivilege(''UniversalFileWrite'');
}
</SCRIPT>
<FORM>
File Name: <INPUT TYPE=''text'' NAME=''file''><BR>
File Data: <INPUT TYPE=''text'' NAME=''data''
ONCHANGE=''save_it(this.form.file.value, this.value)''>
</FORM></BODY></HTML>
```

Die zweite Möglichkeit birgt ein noch weitaus höheres Gefahrenpotential. Im vorherigen Abschnitt wurde bereits ein Codebeispiel gezeigt, welches einen APPLET-Tag an einem Filter durch dynamische Generierung aus JavaScript heraus erzeugt. Filter, die die Möglichkeit haben, Java-Bytecode abzublocken, sind deshalb unbedingt einzusetzen, wenn man differenziert filtern will. Allerdings bietet dies keinen Schutz vor ActiveX. Hier bietet sich keine sinnvolle Möglichkeit, Binärcode abzublocken, so daß die dynamische HTML-Generierung hier ein sehr hohes Gefahrenpotential bietet. Schlimmer noch: ist ein ActiveX-Control bereits installiert, muß gar nichts mehr abgeblockt werden, da es dann direkt ausgeführt wird.

3.5 Datenübertragung und Filterung

Da JavaScript im wesentlichen innerhalb des HTML-Stroms vorkommt, ist zum Abblocken dieser Technologie ein HTML-Filter unabdingbar. Dieser ist in seiner Komplexität nicht zu unterschätzen, da anders als bei ActiveX und Java nicht nur nach bestimmten Tags gesucht werden muß, sondern auch die Attribute innerhalb von Tags untersucht werden müssen. Leider werden von den meisten Programmen, die anbieten, JavaScript zu filtern, nur SCRIPT-Tags und Eventhandler herausgefiltert.

Externe JavaScript-Dateien, die durch ein SRC oder ein ARCHIVE-Attribut angegeben werden, werden in der Regel als Datei mit der Endung ".js" und dem MIME-Type "application/x-javascript" übertragen; dies ist jedoch nur Konvention. JAR-Dateien sind genauso zu behandeln, wie im Javafall. Digitale Signaturen sind hier eine Filtermöglichkeit, sofern sie vorhanden sind. Auch eine Inhaltsanalyse wäre möglich, allerdings läßt sie sich aufgrund der anderen Struktur (ASCII-Dateien verglichen mit "organisiertem" Bytecode) nicht so effizient durchführen.

3.6 Andere Scriptsprachen und Implementationsunterschiede

Microsoft hat noch eine weitere Scriptsprache eingeführt: VBScript. Diese Sprache ist ein Subset von Visual Basic und soll es Programmierern, die bereits Visual Basic beherrschen, vereinfachen, aktive Webinhalte zu erstellen. Es fehlen unter anderem Dateioperationen und einige Operatoren. Unter `http://msdn.microsoft.com/scripting/default.htm?/scripting/vbscript/vbslang/vsgrpNonFeatures.htm` gibt es eine Liste der nicht vorhandenen Eigenschaften.

Die Microsoft-Scriptingengine für JavaScript heißt JScript. Diese umfaßt einen höheren Funktionsumfang als Standard-JavaScript. So gibt es hier z.B. eine Klasse `FileSystemObject` und mit der Version 3.0 (ab IE 4.X) auch `File` und `Folder` Objekte.

4 Schutzmechanismen

Wie schützt man sich nun vor unerwünschtem ausführbaren Code? Die einfachste Möglichkeit ist das Abschalten im Webbrowser. Eine andere Variante ist die zentrale Filterung an einer Firewall zum Schutz eines ganzen LANs.

4.1 Schutz des einzelnen Benutzers

Hier sollen die Einstellungen, die am individuellen Browser konfiguriert werden können, untersucht werden.

Bei den Netscape Browsern können Java und JavaScript nur "An" oder "Aus" sein. Es ist hier nicht möglich, nach Quelle des Codes oder digitaler Signatur zu Filtern. Netscape bietet auch nicht die

Funktionalität, es von einer Benutzerbestätigung abhängig zu machen, ob geladener Code gestartet wird.

Internet Explorer ab der Version 4 ist da schon etwas flexibler. Hier kann mit Hilfe der Sicherheitszonen für die verschiedenen Technologien konfiguriert werden, ob sie bei Herkunft aus dem lokalen Netz oder dem Internet ausgeführt werden sollen, bzw., ob der Benutzer vor jeder Ausführung dies bestätigen soll. Dabei gab es leider eine Sicherheitslücke, mit der man die Einstufung internes/externes Netz austricksen konnte, die allerdings schon durch einen Patch behoben ist.

Für keinen der beiden Browser gibt es die Möglichkeit, so wie bei Suns HotJava, den Java Security-Manager zu konfigurieren, wobei dies mit Netscapes "Grant"-Modell in anderer Form zur Verfügung gestellt wird.

Netzwerkadministratoren haben zwar die Möglichkeit, bei zentraler Softwareverteilung Browser mit 'sicheren' Einstellungen zu verteilen, dies kann aber von Benutzern durch das Ändern der Einstellungen oder eigene Installieren von Browsern unterlaufen werden.

4.2 Zentralisierte Technologien

Die individuellen Einstellungen am Browser sind für geschulte Anwender oft die einzige Möglichkeit zum Schutz. Für den Administrator, dem die Sicherheit eines ganzen Netzwerkes obliegt, ist dies allerdings unzureichend, da es hier nicht um den individuellen Schutz eines einzelnen PCs geht, sondern um den Schutz des ganzen Unternehmensnetzwerkes, und dabei nicht akzeptiert werden kann, daß ein einzelner Benutzer aus Unwissen oder Mutwillen doch etwas anders einstellt.

Deswegen empfiehlt es sich in dieser Situation, die Filterung an einer zentralen Stelle durchzuführen. Da LANs, die an das Internet angeschlossen werden, in der Regel eine Firewall zum Schutz haben [5], bietet es sich an, den Filter in deren Nähe anzuordnen, da alle Webzugriffe über diese Firewall ablaufen sollten. Wie bereits beim Blick auf die Technologien angesprochen ist das Scannen des Inhaltes nicht trivial. Deshalb sollte der Rechner, der diese Aufgabe übernimmt, mit einem ausreichenden Maß an CPU-Kapazität und Speicher in Abhängigkeit von der Leitungskapazität des Internetanschlusses und der Anzahl der Benutzer ausgestattet sein. Die Filterung selbst geschieht im Datenstrom. Der Filter kann entweder ein Proxy sein, der auch im Webbrowser eingetragen ist und die Requests beantwortet, oder aber ein sogenannter "transparenter" Proxy, der den Anschein erweckt, daß der Browser eine direkte Verbindung in das Internet hätte, tatsächlich aber den Datenstrom abarbeitet und so filtern kann.

Ein Problem hierbei stellen natürlich SSL-Verbindungen dar, welche im Regelfall nicht am Proxy ent- und verschlüsselt werden, so daß keine Filtermechanismen greifen können. Hier wird es wohl Produkte geben, die dies auf dem Clientsystem abblocken können, indem sie sich in den Browser einhängen.

Es gibt folgende Möglichkeiten der Anordnung:

Außerhalb der Firewall Auf der dem Internet zugewandten Seite steht ein Rechner, der die Aufgabe der Filterung übernimmt. Dies hat den Vorteil, daß auch auf einem evtl. auf der Innenseite stehenden Proxy mit Cache keine ausführbaren Daten ankommen. Der offensichtliche Nachteil ist die Angreifbarkeit (und damit auch eventuelle Manipulierbarkeit) dieses Rechners.

In der Firewall Jede Firewall ist entweder proxybasierend, paketfilterbasierend oder eine Hybridform dieser beiden Technologien. So bietet es sich also bei Proxies auf einfache Weise an, im HTTP-Proxy Filtermechanismen zu integrieren, die die unerwünschten Inhalte abblocken. Paketfilterprodukte tun sich da etwas schwerer, da sie zunächst einmal erkennen müssen, daß es sich bei einem Datenstrom um HTML handelt, bzw. die Filterung bestimmten Portnummern zugeordnet werden

[5] oder haben sollten!

muß. Der Vorteil ist auch hier, daß bei einem funktionierenden Filter keine unerwünschten Inhalte in das LAN kommen, der Nachteil ist, daß die Firewallmaschine selbst für den einen Dienst HTTP sehr stark aufgebläht werden muß.

Neben der Firewall Eine weitere Möglichkeit stellt ein Prozeß neben der Firewall dar. Dabei übergibt der auf der Firewall zuständige Prozeß die HTML-Daten (bzw. Binärdaten bei ActiveX und Java) an einen eigenen Scan-Prozess, der entweder auf der gleichen Maschine oder einer anderen steht. Der zurückgegebene gefilterte Inhalt wird dann an den Benutzer weitergereicht. Auch hier gelangen keine ungefilterten Daten in das LAN. Der Nachteil hierbei ist der Durchsatzverlust, da alle Daten entsprechend öfter hin und herkopiert werden müssen.

Innerhalb der Firewall Schließlich bleibt noch die Möglichkeit, die Filterung direkt zwischen dem Benutzer und seinem nächsten Verbindungspunkt (dem Proxy oder dem Internet) durchzuführen. Dies hat den Vorteil, daß es auf einem dedizierten Rechner auf der Innenseite des LANs geschehen kann, bedingt aber andererseits, daß der Zugriff auf die Firewall von innen geschützt werden muß, um ein Unterlaufen der Sicherungsmaßnahmen zun verhindern.

Eine weitere Technologie, die nicht so ganz in diese Liste paßt, ist das Ablaufenlassen aktiver Inhalte in einer geschützten Umgebung, wobei die Ausgabe auf den Rechner des Benutzers umgeleitet wird. Wenn die Ausgabeumlenkung in einer sicheren Art und Weise gewährleistet werden kann, ist dies in etwa so sicher wie die Einführung von Surf-Stationen, da auch hier kein Inhalt in das LAN gelangt.

4.3 Komplexität

Wie bei der Untersuchung der bestehenden Filterprodukte noch deutlich werden wird, ist das Untersuchen von Inhalten durchaus rechenintensiv. Performancerückgänge auf ca. 25% des theoretischen Maximums ohne Filter sind hier keine Seltenheit. In diesem Abschnitt soll auf die Problematik des Scannens an sich eingegangen werden. HTML-Dateien sind im ASCII-Format abgelegt. Um etwas Zahlenmaterial zu ermitteln wurden die in den Caches zweier Webproxies enthaltenen HTML-Dateien untersucht. Im Mittel sind sie ca. 6-7k groß, wobei die Varianz ziemlich hoch ist (beim Scannen des Webproxies wurden HTML-Dateien von 50-150000 Byte gefunden). Die Kontrollelemente, die sogenannten HTML-Tags, sind von spitzen Klammern umgeben und enthalten neben dem eigentlichen TAG optional auch Attribute, welche je nach Attribut auch noch Argumente enthalten können. Bei der bereits erwähnten Messung ergab sich, daß der Durchschnittswert der HTML-Tags pro Datei bei ca. 200 liegt, wobei auch hier die Varianz recht hoch ist. Zur Identifizierung eines TAGS und amit der Entscheidung, ob es sich um einen aktiven Inhalt handelt, genügt es, die ersten sechs Zeichen (APPLET, SCRIPT und OBJECT sind gleich lang), die kein Leerzeichen sind und der spitzen Klammer folgen, zu untersuchen. So läßt sich sehr einfach und effizient ein reiner Tagfilter erstellen, der mit 200 Stringvergleichen und 6000 Zeichenvergleichen pro Datei auskommt.

JavaScript vollständig zu scannen, steigert die Komplexität nicht unerheblich. Innerhalb eines jeden Tags müssen die optionalen Attribute und deren Argumente erkannt und mit dem entsprechenden Vorkommen von JavaScript verglichen werden. Außerdem müssen die Tags STYLE und LINK gesondert behandelt werden, da auch sie zu filternden Inhalt besitzen, wenn sie das Attribut TYPE=text/javascript enthalten.

Da HTML kontextinsensitiv ist, müssen sämtliche String und Zeichenvergleiche auch so ausgeführt werden, was die Anzahl der Rechenoperationen verdoppelt.

5 Die Psychologische Komponente

Die Sicherheitsmodelle der beiden Browser basieren darauf, daß letzten Endes der Benutzer eine Entscheidung treffen muß, ob eine Aktion zugelassen wird oder nicht. Um aufzuzeigen, mit welchen Entscheidungen der Benutzer durch Alert-Boxen konfrontiert wird, und welche Konsequenzen dies hat, hier einige Beispiele:

5.1 Grants

Fordert ein Java-Applet oder JavaScript-Code einen Grant an, so erscheint eine Meldung wie die folgende:

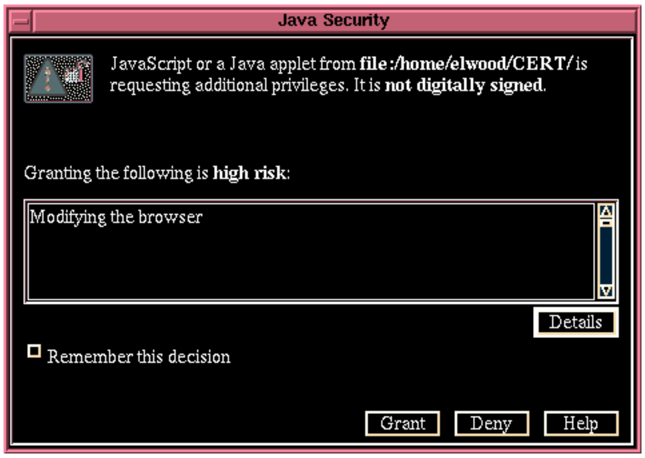

Abbildung 2: Granterteilung bei Netscape

Ein Klick auf den Details-Button ergibt folgenden erklärenden Dialog:

Abbildung 3: Grant Details bei Netscape

Diese Meldungen sind bei machen Grants schon recht aussagekräftig (z.B. unter UniversalFileWrite kann man sich schon etwas vorstellen), aber gerade UniversalBrowserWrite eröffnet mehr Möglichkeiten (z.B. das Ändern von FileUpload-Feldern), als man hinter diesem Namen vermuten könnte. Eine etwas detailgenauere Formulierung dieser Grants wäre wünschenswert.

5.2 Microsoft Internet Explorer

Der Internet Explorer ist ab der Version 4 so konfigurierbar, daß vor der Ausführung eines jeden ActiveX Controls oder JavaScript Inhaltes eine Alert-Box wie die folgende erscheint:

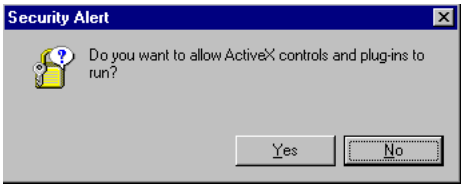

Abbildung 4: ActiveX-Ausführung bei IE 4.0

Dies trifft keinerlei Aussage darüber, was passiert, wenn dieser Code ausgeführt wird, und einmal gemachte Bestätigungen merkt sich der Browser nicht. Immerhin ist der Benutzer so zumindest gewarnt, bevor Code ausgeführt wird. Wünschenswert wären Angaben über die Herkunft des Codes (gleiche URL oder gar ein anderer Webserver), optionale Zertifikate und im Falle von JavaScript für erfahrenene Benutzer eine Button, durch den man direkt und unkompliziert einen Blick auf den Code werfen kann.

Im Falle der Installation von ActiveX-Komponenten wird dem Benutzer eine Box wie die folgende präsentiert:

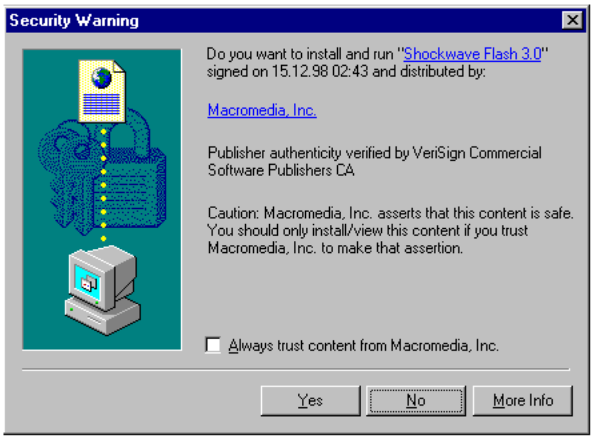

Abbildung 5: Installation von ActiveX Komponenten

Hierbei wird vorausgesetzt, daß dem Benutzer klar ist, was ein Zertifikat ist und wer der Hersteller der Software ist. Es wird jedoch an keiner Stelle auf eventuelle Gefährdungen aufmerksam gemacht.

6 Anforderungen aus der Praxis

Die Ausführungen bis hierhin lassen als Konsequenz die Empfehlung "Alle Inhalte ausschalten und abblocken" als die sinnvollste erscheinen. Der Autor hat schon in einigen Projekten Filtermaßnahmen eingeführt und ist dabei auf einige Problematiken gestoßen.

Eine vollständige Sperrung aktiver Inhalte macht viele Webseiten unbenutzbar. Leider ist bei Webdesignern das Sicherheitsbewußtsein nicht sehr hoch und viele kommen auch gar nicht auf die Idee, daß es viele, vor allem kommerzielle Kunden, gibt, die eben durch eine Firewall kein blinkendes JavaScript sehen können. Leider gibt es auch genug Webseiten, auf denen die Navigation ohne aktive Inhalte nicht klappt, oder wo Teile der Information nicht dargestellt werden. Wenn der Benutzer Glück hat, bekommt er noch eine Meldung, daß man mindestens einen Browser der Version X.Y braucht, oder daß Java oder JavaScript eingeschaltet sein müssen. So entsteht der Druck auf den Systemadministrator, Ausnahmen zu machen[6]. Dies hat zur Konsequenz, daß eine Filtersoftware die Möglichkeit bieten muß, Ausnahmen zu machen. Dies sollte in einer einfachen Form und nach Möglichkeit auch mit Wildcards machbar sein[7].

Ein wichtiger Punkt bei der Filterung nach digitalen Signaturen ist, daß neue CAs zur Liste der als vertrauenswürdig eingestuften Unterschreiber hinzugefügt werden können. Dies sollte eine Selbstverständlichkeit sein, aber die Einbindung von CAs in den Internet Explorer, die nicht von einer CA signiert sind, die dem Browser bekannt ist, bereitet Probleme.

7 Existierende Filterprodukte

Um den Test durchzuführen, wurde das zu untersuchende Filterprodukt auf einem Rechner mit zwei Netzwerkkarten installiert. Beide Seiten standen in einem 10Mbit-Netz. Von einem Client-System aus wurde immer wieder der gleiche Dateibaum auf einem Webserver auf der anderen Netzwerkseite einige hundert mal abgefragt.

7.1 Firewall-1

Firewall-1 bietet die Möglichkeit, zum HTTP-Protokoll sogenannte 'Resources' zu definieren. Dabei existiert ein TAG-Filter für JavaScript/ActiveX/Java und die Möglichkeit, Java-Klassendateien abzublocken. Im Falle JavaScript wird der Script-Tag durch das Ersetzen des Ts durch ein ! unwirksam gemacht und es werden Eventhandler ebenso durch Ersetzen des letzten Zeichens des onXXX durch ein ! unwirksam gemacht. Alle anderen Methoden können den Filter passieren. Der Performanceverlust auf dem Testsystem betrug ca. 62%.

7.2 Safegate

SafeGate der israelischen Firma Security7 bietet einen sehr detaillierten Schutz im Bereich Java. Hier ist es möglich, für Verbindungen aus Client und Webserver zu definieren, ob Applets, die bestimmte Java-Klassen benutzen, passieren dürfen oder nicht. Weiterhin können ganze URLs gesperrt werden, und es gibt eine Liste bekannter feindlicher ActiveX-Controls und Java-Applets, die gesperrt werden können. Für JAR-Archive oder CAB-Files kann auch nach digitaler Signatur oder CA, die diese unterschrieben hat, gefiltert werden. Das Ganze kann auch auf Kombinationen aus Clientsystemen und Zielsystemen in einfacher Art und Weise eingeschränkt werden; wie der Hersteller versicherte wird auch benutzerbasierende Freischaltung in Kürze möglich sein. Der Performanceverlust lag bei 50 %. Im Falle JavaScript werden momentan nur SCRIPT-Tags gefiltert.

[6] Ganz zu schweigen von den Geschäftsführerausnahmen, wenn der Manager voller Stolz die Homepage seines Kegelclubs zeigen will, in der auch JavaScript enthalten ist

[7] Wer z.B. Netscapes Website mit JavaScript benutzen will, sollte lieber *.netscape.com freischalten, da man sonst www, home, support,... einzeln freischalten muß

7.3 SurfinGate

Finjan's Produkt arbeitet als Proxy. Es bietet die Möglichkeit, Java und ActiveX-Controls selectiv abzublocken. Dabei kann dies sogar nach Benutzer oder Source-IP-Adresse unterschiedlich gehandhabt werden. JavaScript wird von den untersuchten Produkten am besten gefiltert, aus dem HTML-Strom wurden SCRIPT-Tags, Eventhandler, Entities und JavaScript-URLS herausgefiltert. Lediglich LINK-Stylesheets, Conditional Comments und STYLEs vom Typ text/javascript konnten passieren. Der Performanceverlust lag bei ca 65%.

7.4 Netscape ProxyServer

Netscape Proxy Server bietet die Möglichkeit, bestimmte HTML-Tags (dabei vordefiniert u.a. APPLET, SCRIPT, OBJECT) herauszufiltern. Dabei kann die Liste um beliebige <TAG>-/TAG Paare erweitert werden. JavaScript kann hier nur in der Variante des Codes innerhalb der Script-Tags gefiltert werden, alle anderen Varianten passieren den Filter unbehelligt. Außerdem ist es möglich, bestimmte MIME-Types abzublocken. Dabei sind diverse Typen für z.B. Java-Archive oder application/java-vm bereits vordefiniert. Da der MIME-Type aber nur Konvention ist, ist dies wenig hilfreich. Der Performanceverlust durch den Tag-Filter betrug ca 29%, bei eingeschaltetem Virenscanner 63%. Freischaltung ist nur durch Regular Expressions in den Ziel-URLs möglich, was bei längeren Listen die Performance dramatisch reduziert.

7.5 TrendMicro Interscan VirusWall 3.1

TrendMicros Interscan VirusWall ist an sich ein Produkt zur Suche nach Viren. Es gibt in neuen Version allerdings die Option, Java-Klassendateien sowie ActiveX-Controls aufgrund der Dateiendung abzublocken. Einen Tag-Filter gibts es nicht, genausowenig wie einen Schutz vor JavaScript. Der Performanceverlust betrug 36%.

7.6 Fazit

Keines der vorgestellten Produkte bietet einen vollständigen Schutz vor JavaScript. Dieses wird von einigen Herstellern sogar vollständig ignoriert. Erste Messungen haben gezeigt, daß bei der Auswahl des Rechners nicht unbedingt gespart werden sollte, wenn eine schnelle Internetanbindung vorhanden ist. Es wurden Performanceverluste im 10Mbit Lan auf ca 30% der Leistung ohne Filterung festgestellt, was aber bei einer 64Kbit-ISDN-Anbindung noch nicht ins Gewicht fällt. Die erzielten Datenübertragungsraten waren selbst für 2Mbit-Leitungen noch hinreichend.

8 Zusammenfassung und Ausblick

Aktive Webinhalte sind Technologien, bei deren Entwicklung wohl mehr auf den schnellen Fortgang des Projekts denn auf den Sicherheitsaspekt geachtet wurde. Dies führte zu konzeptionellen Fehlern, die man jetzt nur noch durch Nachbesserungen oder inkompatible Konzepte versucht aufzufangen.

In der Regel sind dadurch auch die Benutzer für die Problematik nicht sensibilisiert, weil ja "in älteren Browserversionen da auch nix stand".

Einige Punkte, die man sich aus Sicht der Sicherheit wünschen könnte, wären die folgenden:

- Eindeutige MIME-Types, die nicht nur Konvention sind, sondern ohne die ein Browser auf einen aktiven Inhalt nicht reagiert
- Deutlichere Warnung an die Benutzer, damit diese auch wissen, daß sie sich einem Risiko aussetzen
- Für höhere Sicherheitsansprüche bleiben zwei Möglichkeiten:
  1. Die Benutzer ausreichend schulen, do daß sie verantwortungsvoll mit dem Internet umgehen können
  2. Gesicherte Ablaufumgebungen schaffen, in denen böswilliger Code keinen Schaden anrichten kann
- Getrenntes Einstellen der Technologien pro besuchter Website sollte auch im Browser möglich sein.
- Die am Markt erhältlichen Schutzprodukte hinken zum Teil der sehr dynamischen Entwicklung im Internetmarkt hinterher.
- Code aus HTTPs-Quellen wird auf Grund der Verschlüsselung in der Regel überhaupt nicht gescannt, obwohl auch dieses mit entsprechenden Proxies möglich wäre. Dieser birgt ein höheres Gefahrenpotential, da er in den Fähigkeiten Code mit digitalen Signaturen gleichgestellt ist.

Literaturverzeichnis

[1] David Flanagen, *JavaScript: The definitive Guide, 3rd Edition*, O'Reilly & Associates, 1998

[2] Musciano & Kennedy *HTML: The definitive Guide, 2nd Edition*, O'Reilly & Associates, 1997

[3] ??, *Privacy for Browser Users*, MSDN Fall 1998

[4] Neurotec, *OCOCAT-S*, Bundesamt für Sicherheit in der Informationstechnik, 1997

[5] Dr. Kai Fuhrberg, *Neue Entwicklungen im Bereich Intranet und Internetsecurity*, Neues aus dem BSI, Bundesamt für Sicherheit in der Informationstechnik, 1998