

# Keeping the Genie in Your Bottle: Taming Active Content on the WWW

Alec Muffett  
Senior Staff Engineer  
Sun Microsystems Laboratories  
Alec.Muffett@UK.Sun.COM

All trademarks referenced in this document are held by their respective owners

# Active Content

- What is it?
- How does it relate to the WWW?
- What happens when you use it?
- How has it come about?
- What different types exist?

# Active Content

- What are the threats?
- What are the solutions?
- What technologies can help you?
- How safe can you be?

## Goal

At the end of this seminar, you should be able to answer:

- Is Active Content a genie in a bottle?
- Or is it a can of worms?

... for yourself, given your own circumstances.

## Things we will assume

- You have a rough idea of how the WWW works.
- You have used a fairly modern browser.
- You have seen some of the hype regarding Active Content.
- You want to know more.

# Chapter 1

## What is Active Content?

# What is Active Content?

Active Content (AC) is data, retrieved from or sourced by the net, which exhibits dynamic behaviour on the client system that receives it.

# What is Active Content?

## In Short

AC intrinsically utilises computational resources when downloaded or viewed, as part of its purpose.

# How has AC come about? History of Active Content

# Early HTML Documents

In the beginning, there was:

- Traditional retrieval of plain text or hypertext, containing formatted information and perhaps some graphics.

## HTML Document Retrieval

Of course, the only activity that such documents may initiate is inside the user's head, although that *may* be enough to compromise security.

# HTML Document Retrieval

This is because people are the weak links in security.

Ask any con-artist.

## Early AC Helper Applications

HTML was not enough; document types were extended to support arbitrary applications.

- Start of serious AC-style danger
- CERT advisory CA-95:10 “GhostScript”  
led to `-dSAFER` option for PostScript interpreter
- Start of “sandbox” concepts seen later in Java

## Helper Applications

- Other threats include definition of MIME-types such as `x-sh` & `x-csh`.
- Yes, people really *will* run obscure code that is handed to them on a plate, because:

“Security is Someone Else’s Problem”

# CGI Form Submission

Server Interaction: the Common Gateway Interface

- Solicit information from the user from within the browser environment
- Information is returned to the server and actioned

## CGI-based Security Problems

- Data returned to server in cleartext over network
- No adequate authentication framework was available
- Poorly-written CGI “backend” programs misbehave when supplied with malicious input.
- No guarantee of secure storage on server

# CGI Problems

## Cleartext Transmission

Some people seem to believe if a wire is covered in plastic, you cannot see what is travelling along it; alas for:

- Administrators running proxy hosts and firewalls
- Administrators on intermediate nodes/routers
- Hackers running network sniffers
- Anyone who is interested along the route

# CGI Problems

## Weak Authentication

Early WWW authentication was little more than passing:

*username:passwordstring*

... along with any requests for document retrieval that you might make, in cleartext; this is open to:

- traffic sniffing
- an infinite number of replays by other users
- attempts to re-use *passwordstring* in other contexts

# CGI Problems

## CGI-backend hacking

Badly written programs are susceptible to malicious data:

- CGI program reads a *hostname* from the network
- CGI program then spawns “finger @*hostname*”

but if: *hostname* = ‘rm -rf /’

... the server will be trashed  
if the CGI backend is badly written.

# CGI Problems

## Secure Storage

If you know where the important information is stored *en-masse* (ie: the WWW server) it makes a juicy target for a hack attack.

## Intermediate AC Plug-ins

- Similar to helper applications.
- Typical “commercially” sourced.
- Intimately bonded with the browser environment.

## Intermediate AC Plug-ins

- Plug-ins are popular with software houses as they extend a market niche.
- Depending on what the plug-in tries to do, threats are variable, but similar to Helper Apps.
- There may be quality/integration issues with third-party plug-in vendors.

## Fully Active Content

Executable code embedded in HTML (scripting) or separate modules (Java, ActiveX) loaded and processed by an interpreter or by the CPU.

## Fully Active Content

This stuff is a major threat unless you:

- have some control over actions the code may (or may not) perform.
- know that the code came from a trustworthy source
- know that the code has not been meddled with.

...or (preferably) all three; more of this anon.

## Chapter 2

What happens when you use  
Active Content?

# What happens . . . When you use AC?

Possibly the best way to answer this is to look at what happens when you retrieve *any* document.

# What happens . . . When you retrieve a document?

- What are you actually doing?
- How sure can you be of where you got it from?
- How much faith can you place in its content?

# What are you doing?

You are retrieving data from a net-connected remote site;  
one which purports to have some particular name.

# What are you doing?

You are trusting that ...

- the name of the site maps to a organisation you know.
- the site is run by (or for) that organisation.
- the site maintains good integrity.
- the data you retrieve will not harm your system.

# Where did you get it from?

“an organisation you know”

- <http://www.csi.com/> is not *this* CSI ...
- <http://www.micros0ft.com/>
- <http://www.british-telecom.com/>
- “claim-jumping” domain names
- security failures in hostname mapping

# Faith in its Content

“maintains good integrity”

You assume that your investment in *time spent downloading data* will yield worthwhile results.

# Faith in its Content

“maintains good integrity”

Examples:

- Site may have data stolen (eg: data about *you*)
- Site has been lampooned  
(eg: CIA, DoJ, UK Labour Party)
- Site has had data modified  
(eg: price lists, AC you might download)

# Faith in its Content

“maintains good integrity”

Subtle modifications to a site's content could lose:

- money from litigation
- corporate credibility
- customers
- votes

## Where the WWW loses “not harm your system”

- Wasted working time  
(staff “surfing the net”)
- Potential download of viruses or worms  
(overhead of repairing damage)
- Potential execution of malicious AC  
(loss of privacy, integrity, time, money, etc.)

Where does the WWW lose out?

Fundamentally, it loses out in  
issues of “trust” .

## In Whom Do You Trust?

When you retrieve a document, you trust:

- Your mind (is the URL correct? complete?)
- Your browser (taking you to the right place?)
- Your proxy-cache (not returning out-of-date data?)
- The Domain-Name Service  
(looking up the right IP address?)

*(more)*

## In Whom Do You Trust?

When you retrieve a document, you also trust:

- Your network (routing you to the right place?)
- Your ISP and their peers (not snooping on you?)
- The remote site (being run by whom you expect?)
- The author of the data (accurate? up to date?)

## In Whom Do You Trust?

Once a document is downloaded, what do you do?

- read it
- act upon it
- execute it (AC)

... and if your trust is misplaced, you pay the price.

## In Whom Do You Trust?

The miracle is that retrieval usually works as expected,  
because it is what people *want*.

# In Whom Do You Trust?

The problem is that its working  
is neither foolproof nor assured.

## How do we address this?

The same way we try to do in the real world:

- Defend our own property (Firewalls)
- Establish secure lines of communication (SSL, Cryptography, etc)
- Establish “trusted” parties who can vouch for identity (Certificate Authorities)
- Establish sets of permitted/forbidden behaviour (Policy, Technology)

## Real Life

Problem is, we live in a suck-it-and-see Internet; our options often are reduced to:

- Run it, and accept the possible consequences.
- Don't run it, and risk missing out.

Often, there is no real alternative.

# Chapter 3

## Active Content Types?

# Types of Active Content

There are a lot of these ...

## AC Technologies

- HTML • Dynamic HTML • VRML • RTF • Java • JavaScript • {ActiveX, DDE, OLE, COM, DCOM, DNA}
- VBScript • CORBA • Plug-ins (eg: Shockwave) • Helper Apps (eg: PostScript) • Push Technologies (eg: Pointcast) • Animated GIFs • Raw Binary Code • Plausible User Dialogues - CGI, HTML, ASCII •

Really, anything you can drive from a browser.

## General Forms of AC

- Retrieved – traditional “fetch” paradigm.
- Delivered – subscribed to, or is pushed; includes E-mail/USENET.
- Infectious – viruses, Internet worm, etc . . .

# Chapter 4

## Active Content Function?

## What does AC do?

To review, AC is typically used to:

- extend browser functionality to new technologies
- extend browser connectivity and network integration
- dislocate clients from requiring a server for all functions
- integrate browser with other applications software
- provide faster, more flexible user interaction

...from a single user interface.

# What does AC do?

Examples:

- real-time interaction in user-interfaces
- support for new multimedia and presentation formats
- connecting familiar browsers into databases
- integration with specialised document viewers and other utilities
- eradication of stateful CGI programs for complex form-filling

## How is AC transported?

AC typically arrives in your system by one of three routes:

- individual modules retrieved by a user
- scripts embedded in other document content
- encoded modules attached to some other data

## How is AC transported?

Examples:

- Applets retrieved over the WWW
- JavaScript code embedded in HTML
- Macros embedded in word-processor documents
- Viruses embedded in PostScript (or similar)
- Executable MIME attachments in E-mail or USENET

## How is this different from traditional viruses?

Perhaps the only differences are that viruses:

- are purely malignant.
- are usually associated with binary executables rather than glorified text-files.
- reproduce themselves to infect other hosts.

...but the similarities are worth thinking about.

What kinds of AC  
am I likely to encounter?

first, some clarification

# Marketing Wars

## Java vs. ActiveX

- Java, JavaScript, VBScript, etc  
= Active Content Technologies
- Applets, Plug-ins, ActiveX  
= Packaging/Integration Technologies

# Marketing Wars

## Java vs. ActiveX

- ActiveX is a shrinkwrapping method to contain executable content
- ActiveX competes with Applets (ie: shrinkwrapped Java) and Netscape Plug-ins (ie: shrinkwrapped DLLs).

# Marketing Wars

## Java vs. ActiveX

- ActiveX controls can even be *written* in Java ...
- ... but the benefits of Java security model may be lost if “native” code is used ...
- ... or if configuration data is external to the control.

# Marketing Wars

## Java vs. ActiveX

ActiveX does not compete with *Java*  
except in the minds of marketing creatures.

# Deobfuscation

## ActiveX, October 1997

“ActiveX” has been re-marketed as “COM” by Microsoft.

Note the following terms:

“OLE” “NetDDE” “COM” “COM+”  
“DCOM” “Windows DNA”

... and check what this technology is called when you review these notes; for convenience we shall refer to this technology throughout as “ActiveX” and not “COM”.

# Deobfuscation

## JavaScript is not Java

- Java is a byte-compiled, object-oriented, interpreted language from Sun, etc . . .
- JavaScript (aka: LiveScript) is a scripting language embedded in HTML documents

The two languages are unrelated other than some commonalities of syntax, and that they are both used to implement AC.

## Comparison

What distinguishes the market leaders?

## AC Comparison

### Plug-ins vs. ActiveX vs. Applets

- Plug-ins: typically implements drivers for whole new document types.
- ActiveX: typically implements features, widgets, or applications.
- Applets: typically implement features, widgets, or applications.

## Comparison

### Plug-ins vs. ActiveX vs. Applets

- Plug-ins: semi-manual installation, becoming more automated.
- ActiveX: was: default promiscuous auto-installation – has improved since.
- Applets: default sandbox execution with restricted rights – configurability variable upon implementation.

## Comparison

### Plug-ins vs. ActiveX vs. Applets

- Plug-ins: digital signatures in Navigator v4  
“JAR files”
- ActiveX: digital signatures  
“Authenticode”
- Applets: digital signatures  
“JAR files”

# Comparison

## Plug-ins vs. ActiveX vs. Applets

- Plug-ins: raw libraries (C++) glued into browser process environment.
- ActiveX: binaries (C++), scripts (VB), Java, etc... with unrestricted access to the OS.
- Applets: JVM code executing with restricted privs, JIT compiled on some platforms.

# Chapter 5

## What are the threats from Active Content?

## Basis of Threat from AC

AC is not completely under your/the administrator's control.

## Business Considerations

Before you start, do consider:

- Is AC useful to your work?
- Is AC necessary for your work?
- Is AC unavoidable at your work?
- Is AC a part of your security policy?

Is your journey really necessary?

# Business Considerations

After all, disconnection may be a viable solution.

## Threat Types

Potential for loss of integrity:

- Denial of Service (personal? network-wide?)
- Loss of Privacy (customer lists?)
- Loss of Secrecy (trade secrets?)
- Loss of Data (customer orders? money transfers?)

## Threat Types

### “Denial of Service”

- Animated GIFs and page reloads can thrash your CPU to death.
- Window creation can swamp system resources, crash systems.
- Hostile AC could monitor/kill competing applications from other vendors.

## Threat Types

### “Denial of Service”

- Denial of Service attacks waste time, waste money.
- If *lots* of users are affected, they waste *lots* of time, and waste *lots* of money.

# Threat Types

## “Loss of Privacy”

Bugs that have occurred, non-exclusively:

- Files on your hard-disk could be read (Shockwave)
- List of sites you access on WWW could be mailed to a remote site (JavaScript)
- Arbitrary code could be executed (some Java implementations)

*(more)*

## Threat Types

### “Loss of Privacy”

- HTTP “Referrer” links leak information on where you’ve been previously
- E-mail addresses from WWW/USENET are being sold to “Spammers”
- “Registration” software indexing your hard-disk

# Threat Types

## “Loss of Secrecy”

You could lose:

- Customer lists
- Price lists
- Competitive advantage

... potential for loss depends upon how much you value the secrecy of your data.

## Threat Types

### “Loss of Integrity”

- Live system could be shut down  
(“Internet Exploder” ActiveX Demo)
- Money transfers made  
(“Chaos Computer Club” ActiveX Demo)
- Bogus IRS tax-returns submitted  
(JavaWorld ActiveX Demo)

# Threat Types

## “Loss of Integrity”

No coincidence that previous slide featured ActiveX heavily.

Active Content threat is proportional to what checks you have on the activity that AC can perform.

ActiveX can do a lot, without restriction.

# Threat Vectors

- Trojan Horses
- Broadcast Attacks
- Human (eg: Chain E-mails)
- Random software downloads
- Homogeneity
- Complacency
- Malignant Content

# Threat Vectors

## Trojan Horses

- Even plain ASCII is active, if it is persuasive enough to fool people into doing something.

# Threat Vectors

## Trojan Horses

- Spoof user expectations: make users surrender their passwords to a plausible-looking request.
- Confidence-trick dialogues: make browsers lie about what the user is looking at.
- Forge user-sourced traffic: create bogus purchase orders, embarrassing “Kill the President!!!” E-mails ...

# Threat Vectors

## Trojan Horses

- These almost certainly imply malicious intent . . .
- More likely to be an internally-sourced threat?

# Threat Vectors

## “Broadcast” Hacking

- Hackers no longer need to be deep; they just replay exploits blindly.
- Broadcast an attack (via USENET, E-mail) and it will affect *someone*.

# Threat Vectors

## “Broadcast” Hacking

- These tend to imply malicious stupidity . . .
- More likely to be an externally-sourced threat?

# Threat Vectors

## Human

Wastes of Bandwidth:

- *Good-Times* and *Penpal-Greetings* E-mail Viruses
- “Spam” E-mail
- Chain E-mails

# Threat Vectors

## Downloads

- Software defaults tend towards minimum security & maximum convenience. (eg: ActiveX Controls, Pointcast updates, Plug-in downloads)
- Permitting casual update procedures without verification can lead to disaster
- Administrators lose track of “what’s out there”

# Threat Vectors

## Homogeneous Environments

Browsers extending new functionality to old protocols is a bad thing:

- MIME & HTML over E-mail & USENET ...
- Leads to AC over E-mail & USENET ...
- Leads to insecurity & explosive bandwidth growth.

# Bandwidth

Major UK ISP: 200 staff on well-administered machines.

One day, everything stopped. Cause?

Staff member had posted a GIF attachment to “all” and filled the /tmp and swap-space of four main servers.

# Threat Vectors

## Homogeneous Environments

- Homogeneity leads to casual familiarity
- Familiarity leads to problems  
(cf: “flying a kite” bank thefts)

# Threat Vectors

## Homogeneous Environments

- We no longer require viruses to be attachments; broadcast hyperlinks instead.
- This probably changes legal footing, too.

# Threat Vectors

## Familiarity, Complacency, Ignorance

- More people are encountering more and varied forms of AC every day; eg: MIME/HTML over USENET News and E-mail.

# Threat Vectors

## Familiarity, Complacency, Ignorance

- These messages have clickable attachments and magic headers.
- Average users are nervous and blindly trusting.
- You must educate these people before you have security.

## Cleaning Up After Security Incidents

- Every incident is personal.
- Incidents are most painful to those who have to clean up the mess, afterwards.
- To avoid personal pain, avoid incidents.

# Security Incidents

## The Threat from the Net

- “it’s not as bad as it appears” .
- “no, it’s worse”

# Security Incidents

## The Threat from the Net

- don't be an ostrich.
  - don't think "it'll never happen to li'l-old me".
- ...to do so is to court disaster.

# Security Incidents

“if you are a specific target,  
you cannot find safety in numbers”

Taken from WS&C.

Chapter 6  
Solutions to Threats from  
Active Content ?

## What are the Solutions?

There is no universal solution to security  
and integrity issues of AC.

Instead you must craft a policy to define what is  
acceptable, and then match technologies to this policy.

# AC Policy

## Good Policy Background

- Provide Good Service.
- Platform Standardisation.
- Firewalling and Fireridging.
- Maintain Good Hygiene.

# AC Policy

## Provide Good Service

- Well-Supported Services  
Means users less inclined to “DIY”.
- Up-to-date Services  
Means users experiment less.
- Robust Service  
Means users are happier to accept policy restrictions.

# AC Policy

## Platform Standardisation

- Promote supported, common hardware environment.
- Promote supported, common software environment.
- Promote supported, common user and browser environments.

# AC Policy

## Platform Standardisation

- Simplifies provision of good service, means you have a good idea what is going on in your network and can fix problems rapidly when security alerts are raised.
- Cheaper to license, too.

# AC Policy

## Platform Standardisation

- Yes, this increases threat through homogeneity, but the admin benefit is enormous.
- Do ensure that your *security systems* are varied and diverse . . .
- . . .so long as they can still be understood.

# AC Policy

## Firewalling and Fireridging

- Enforce strong perimeter security (Firewalling)
- Consider partitioning of your Intranet (Fireridging)
- Consider proxy controls  
(Screening, Checking, Stripping)

# AC Policy

## Firewalling and Fireridging

- Restrict inbound traffic through a firewall
- Restrict outbound traffic so that malignant AC cannot easily “call home”
- “Support only that which is necessary”

# AC Policy

## Firewalling and Fireridging

- Consider partitioning-off “sensitive” hosts on your internal networks
- Note that this may have a significant impact on your corporate culture if you are zealous.

# AC Policy

## Maintain Good Hygiene

- Enforce strong host security.
- Maintain an informed userbase.

## AC Policy

### Strong Host Security

- If your host security is good, hackers and malignant AC alike will have a hard time getting anywhere.
- Take a bottom-up approach to security, starting with file permissions on user accounts, and working up to the system/network level.

# AC Policy

## Point for Consideration

Every time you upgrade your browser (which you must do to get the bugfixes you want) ask:

- “Are you aware of what new technologies that you are implicitly taking on-board?”
- “Is your security policy tracking these changes?”

## AC Policy Common-sense Fixes

- Switch off automatic installations/updates if possible
- Examine other “default” settings of your software, comparing to your security policy

## AC Policy Reminder

- Many “denial of service” attacks are *legitimate* functions gone awry, hence the possibilities of even Java being used to implement them.
- See “Hostile Applets” in Java Security book.

## Ideas for controlling . . . an over-enthusiastic boss

- Form a committee to update the security policy.
- Cite browser licensing issues for commercial usage.
- Cite threats to the stock price/company reputation, or similar, “if a disaster happens” .

## Ideas for motivating . . . an under-enthusiastic boss

- Cite saving from not training people for different user interfaces on different platforms.
- Use buzz-words like “Intranet” .
- Buy him/her a subscription to WiReD.
- Get him/her a new machine that can cope.

## Policy Checklist

- If you have no security policy, write one.
- If you have a security policy, revise it. Re-revise annually/bi-annually. Track technology.
- Enforce this policy (*ie: spend money on it, hire or contract-in expertise, audit*)
- Ensure that people know what is (and is not) allowed.
- Make people responsible for their actions.

# Policy Checklist

Know thy enemy:

- Define what software you will permit people to use, to access the WWW
- Track updates to this software for security fixes and new functionality
- Track changes to functionality incumbent in said updates
- Lay in defenses against unrestricted traffic flow, inbound and outbound

## Chapter 7

What technologies can help you?

Firstly . . .

What basic security does AC provide?

# Basic AC Security Technologies

Broadly split into two halves:

- “Sandbox” Model
- Digital Signatures

# Philosophy of Sandboxes

Applications retrieved from an untrusted source should be executed in an environment that prevents them from doing more than is necessary to fulfill their function.

## Java Applets

- Java Application downloaded from WWW server.
- Launched in sandbox under JVM embedded in the browser.
- Activities that may compromise system security are monitored and checked against policy.

# Sandboxes

## Benefits of Sandbox Approach

- Stops users' fingers getting burned
- Random applications can be moderately safely executed; "try before you buy" on the WWW
- Detailed specification of Java yields cross-platform portability

# Sandboxes

## Fine-grained control

New directions in access control:

- Tie sandbox restrictions to application types, eg: games, spreadsheets, wordprocessors . . .
- Ease restrictions where digital signatures of applets augment trust

# Sandboxes

## Problems of Sandbox Approach

Assuming we are talking Java:

- Tracking JVM per-implementation bugs
- Legitimate functionality can still be redirected into a Denial of Service attack
- Threads going on in the JVM background
  - good place to hide something nasty

# Sandboxes

## Summary

Sandboxes are neat functionality.

Of themselves they are not a complete solution,  
but they make a very good start.

# Philosophy of Digital Signatures

“If you can’t control the code, at least you should know where it came from, so you can sue if it goes wrong.”

This has obvious appeal in a country as litigious as the USA.

But . . .

## The Truth of Digital Signatures

Adapted from WS&C:

“Signed code is not [necessarily] safe code”

Comparison with real life:

“Signed cheques [sic] may still bounce”

## Benefits of Digital Signatures

If you can check certificates in a trusted manner:

- You can be sure where the code came from.
- You can be sure the code has not been tampered with.
- You can use an arbitrary (local) download server, because you are sure of the original source.

## Failures of Digital Signatures

Regardless of being signed, digitally-signed code may:

- contain viruses.
- contain worms.
- contain exploitable coding bugs. (eg: stack-overflow)

*(more)*

# Failures of Digital Signatures

Digitally-signed code may:

- remove audit trails
- misbehave if it uses a untrusted configuration file
- do anything you permit it to get away with

Also, the transitory nature of AC makes incident postmortems very painful, especially if there are no constraints.

## Failures of Digital Signatures

If they are not checked (and honoured) automatically, signatures are a fruitless technology.

If they *are* checked and honoured, they add *assurance*, which is a very good thing to have.

# Digital Signatures

## Summary

- Do users really understand the issues or care about digital signatures?
- Is there enough reliable certification infrastructure to support them properly?
- Pity the small user with no funds for legal recourse.

What other tools can help?

# Tools

A plethora of tools are available to filter traffic:

- virus checking
- traffic header stripping (SMTP, HTTP)
- traffic body stripping (SMTP, HTTP, FTP)
- traffic source screening (packet filtering)
- network logging

# Traffic Filtering

AC is delivered to the user in aggregate with passive content, making it hard to detect, filter, or remove.

Nonetheless software to do this, exists.

# Traffic Filtering

Examine whether you can/should utilise (proxy) software that filters traffic:

- by preventing particular types of traffic from being retrieved
- by scanning traffic for particular forms of data
- by preventing data arriving from particular sites

# Traffic Filtering

## Host Blocking

All firewalls should be able to block the arrival of traffic from specified sources; some will even permit administrators to do this at a web-application (URL) level.

# Traffic Filtering

## Type Blocking

Filters are usually based around refusing downloads of filenames with particular MIME types or file suffixes:

`.class, .ocx, .zip, .exe, .jar`

# Traffic Filtering Type Blocking

## Benefits:

- Simple and cheap to implement.
- All or nothing approach simplifies architecture.

# Traffic Filtering Type Blocking

## Problems:

- Keeping up to date.
- Can't catch everything.
- Rather coarse and may block innocent traffic.
- Doesn't catch SCRIPT AC.

# Traffic Filtering

## Traffic Scanning

Filters are usually dependent upon detecting and removing  
AC-related HTML tags in traffic:

SCRIPT, OBJECT, APPLET, EMBED

. . . or similar, eg: detection of virus signatures.

# Traffic Filtering Traffic Scanning

## Benefits:

- Effective for most forms of AC.
- Can be extended to (eg) virus-scan E-mail attachments.
- Provides great audit trails and statistics.

# Traffic Filtering

## Traffic Scanning

### Problems:

- Keeping up to date.
- Can't catch everything.
- Extremely resource hungry.
- Interferes with secure-transport architectures.

# Traffic Filtering

## Sanitising Outbound Traffic

How important is secrecy to you?

- Some tools (even stock software) can hide the structure of your internal network by stripping non-essential information from headers of E-mail and USENET postings, whilst maintaining connectivity.
- Is this a consideration?

## Environment Chokes

Tools that will restrict the environment which AC can affect, by implementing dynamic sandboxes:

- Java Applets do this automatically (“sandbox”)
- Helper Apps *may* be restricted (“janus”)
- ActiveX (“COM”, etc ...) cannot do this?

## Traffic Recorders

Tools that will allow you to reconstruct the flow of traffic around a network in a post-mortem fashion, and perhaps detect unwanted traffic's passage in real time.

## Tool Summary

- You can install technologies that will (within limits) help you control Active Content on your networks.
- You *must* first decide on your policy of what you will permit, so you can select the tools you need.
- Policy first, Technology second.
- The rest, is easy.

## Chapter 8

### How safe can you be?

As safe as you want to be.  
It's for you to take up the challenge.

# Summary

## Reasons to forbid AC

- it can impact security.
- it can impact privacy.
- it can impact integrity.
- it does stuff you cannot expect.

# Summary

## Reasons to permit AC

- it looks funky.
- it is moving from bleeding-edge to core function, though it is unlikely to replace HTML.
- it does stuff HTML cannot.

# Summary

## Personal Viewpoint

- All AC pokes around with your machine somewhat
- I trust Java more than the rest, but not too much, due to implementation fears.
- I am biased because I don't trust other programmers.
- Not even when they sign their code.
- OK, so, I'm paranoid. 8-)

FIN

# Appendix

## Other Reading Books

- Web Security & Commerce; Garfinkle & Spafford; O'Reilly
- Web Security Sourcebook; Rubin, Geer & Ranum; Wiley
- Java Security; McGraw & Felten; Wiley
- Not Just Java; van der Linden; Sun/Prentice-Hall

## Other Reading Books

- Internet Security: Stories from the Trenches; McCarthy; Sun/Prentice-Hall
- Firewalls & Internet Security; Cheswick & Bellovin; Addison Wesley
- Practical Unix & Internet Security; Garfinkle & Spafford; O'Reilly

## Other Reading Resources

- USENET: Security, Java and Infosystems newsgroups
- Secure Internet Programming:  
<http://www.cs.princeton.edu/sip/>
- Javasoft: <http://java.sun.com/>
- Microsoft: <http://www.microsoft.com/activex/>
- Draft security policies - "Site Security Handbook"  
RFC1244 etc
- Janus: <http://www.isaac.cs.berkeley.edu/~daw/janus>

## Useful URLs

- <http://www.digicrime.com/>
- <http://www.halcyon.com/mclain/ActiveX/>
- <http://www.iks-jena.de/mitarb/lutz/security/activex.en.html>