Results of the Security in ActiveX Workshop

Pittsburgh, Pennsylvania USA
August 22-23, 2000

Published at the
CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

December 21, 2000

Table of Contents

Contributors

The ideas in this paper were jointly developed by participants in the Security in ActiveX Workshop. Their intellectual contributions and their spirit of cooperation made the workshop a success. Among the many participants who contributed to this paper are the following:

Steven M. Bellovin
AT&T Labs Research

Cory Cohen, Jeffrey Havrilla, Shawn Hernan,
Brian King, Jeffrey Lanza, Linda Pesante, Rich Pethia
CERT® Coordination Center

Sean McAllister
German Henault
DoD CERT

R. Tom Goodden
DPA, Office of the Under Secretary of Defense (Readiness)

A. Padgett Peterson, P.E. CISSP
Corporate Information Security Office
Lockheed Martin Corporation

Sean Finnegan
Microsoft Corporation

Ken Katano
National Security Agency

Richard M. Smith
Chief Technology Officer
Privacy Foundation

Ralph A. Lowenthal
SYTEX, INC, in support of
the Secretary of the Army, Director of Information Systems
Command, Control, Communications, Computers, and Intelligence (DISC4),
Information Assurance Office

Results of the Security in ActiveX Workshop

Pittsburgh, Pennsylvania USA August 22-23, 2000

Executive Summary

On August 22-23, 2000, the CERT® Coordination Center hosted a workshop in Pittsburgh, Pennsylvania, for twenty invited experts to address security issues related to ActiveX controls. The primary goal of the workshop was to identify the situations under which ActiveX and related technologies may be used safely and to produce a paper describing security concerns and configuration guidance.

That goal was achieved and the result of the workshop, this paper, serves not only to dispel unwarranted myths about the safety of using ActiveX but also to furnish guidance to network administrators and others faced with security issues involving mobile code in general and ActiveX in particular. ActiveX and similar mobile codes provide enhanced usability. The level of enhancement is significant enough for corporate and government users that Internet security policies and procedures should reflect "risk management" rather than "risk avoidance."

Part 1 of this paper provides an overview of ActiveX, including security concerns and security features. Following this general information are, in Part 2, suggestions and good practices for specific groups in the Internet community:

- managers
- system administrators and security personnel
- developers of ActiveX controls and software that uses them
- users who administer their own computers; anyone who doesn't have a system administrator or security expert managing their system

The workshop participants hope that the information offered here will help readers make informed decisions about the security tradeoffs related to ActiveX controls and ultimately improve security in their use.

The contributors to this paper encourage readers to distribute their paper widely. Readers should also be vigilant, keeping informed about further developments, checking the references listed in Appendix C, and monitoring web sites such as those of the CERT Coordination Center (www.cert.org) and Microsoft (www.microsoft.com/security/).

Results of the
Security in ActiveX Workshop

Introduction

On August 22-23, 2000, the CERT® Coordination Center hosted a workshop in Pittsburgh, Pennsylvania, for twenty invited experts to address security issues related to the use of ActiveX controls. The primary goal of the workshop was to identify the situations under which ActiveX and related technologies maybe used safely and to produce a paper describing security concerns and configuration guides.

Part 1 of this paper provides an overview of ActiveX, including security concerns, benefits, and security features. Following this general information are, in Part 2, suggestions and good practices for specific groups in the Internet community:

- managers (p. 11)
- system administrators and security personnel (p. 12)
- developers of ActiveX controls and software that uses them (p. 24)
- users who administer their own computers; anyone who doesn't have a system administrator or security expert managing their system (p. 30)

Finally, three appendices provide a) contact information for reporting vulnerabilities in ActiveX controls; b) a list of known vulnerabilities; and c) sources of further information.

The workshop participants hope that the information offered here will help readers make informed decisions about the security tradeoffs related to ActiveX controls and ultimately improve security in their use.

We encourage readers to distribute this paper widely and to monitor web sites to keep informed about further developments.

PART I: General Information About ActiveX Security

1. ActiveX Overview

ActiveX, or more properly *ActiveX control*, is Microsoft's term for a particular kind of software based on the Component Object Model (COM). ActiveX controls are highly portable COM objects, used extensively throughout Microsoft Windows platforms and, especially, in web-based applications. COM objects, including ActiveX controls, can invoke each other locally and remotely through interfaces defined by the COM architecture. The COM architecture allows for interoperability among binary software components produced in disparate ways.

ActiveX controls can also be invoked from web pages through the use of a scripting language or directly with an HTML OBJECT tag. If an ActiveX control is not installed locally, it is possible to specify a URL where the control can be obtained. Once obtained, the control installs itself automatically if permitted by the browser. Once it is installed, it can be invoked without the need to be downloaded again.

ActiveX controls can be signed or unsigned. A signed control provides a high degree of verification that the control was produced by the signer and has not been modified. Signing does not guarantee the benevolence, trustworthiness, or competence of the signer; it only provides assurance that the control originated from the signer.

ActiveX controls are binary code capable of taking any action that the user can take. ActiveX controls do not run in a "sandbox" of any kind. Because of this, it is important to have a high degree of trust in the author of the control. (This is true for any code that runs directly on your machine and not in a sandbox.)

The security issues relating to ActiveX cannot be ignored. ActiveX controls are an integral part of systems and applications, and they are required for essential functions in many environments. Though priorities many change from organization to organization and user to user, it is important to understand the tradeoffs between functionality and security and to make informed decisions about the appropriate level of risk.

To help readers make those well-informed decisions about Active X controls, the next sections

- clarify some misconceptions
- discuss security concerns and risks
- describe the benefits that prompt tradeoff decisions
- list security features that are available with ActiveX

2. Misconceptions About ActiveX

ActiveX controls present legitimate and significant security concerns, but there are many misconceptions about them. In order to make decisions about risk, it is important to objectively evaluate the risks without hyperbole, exaggeration, or sweeping generalizations. In this section, we clarify the most common misconceptions about ActiveX.

Misconception #1 All ActiveX controls are equally dangerous.
The purposes and quality of ActiveX controls vary widely, just as any type of software. There is nothing inherent in an ActiveX control that necessarily presents a security risk, though, as we discuss later, it may be difficult for system administrators to evaluate the risk presented by a given ActiveX control.

Misconception #2 ActiveX controls are different from "regular" .EXE files; alternately, ActiveX controls are the same as "regular" .EXE files.
ActiveX controls share many attributes with ordinary executable files. They run directly on your hardware; they are generally opaque; they are written in a variety of high-level languages; and they vary substantially in functionality, quality, and security characteristics. However, unlike ordinary executable files, ActiveX controls can be mobile and can often be invoked remotely.

Misconception #3 The greatest security danger presented by ActiveX controls are intentionally hostile controls like Internet Exploder.
It is true that intentionally malicious ActiveX controls can potentially do a great deal of damage. However, in practice very few overtly malicious controls exist, and Microsoft's Authenticode technology provides as much protection from intentionally malicious controls as can reasonably be expected. Of significantly greater concern is the "repurposing" of legitimate controls by attackers through vulnerabilities in those controls.

Misconception #4 All signed controls are safe controls.
Nothing could be further from the truth. Microsoft's digital signature technology, Authenticode, provides a high degree of verification that the control was produced by the signer and has not been modified by an interloper. A digital signature does not, however, provide any guarantee of benevolence or competence. It is up to the end user to decide if a given control from a given producer is trustworthy. And the decision on trustworthiness involves two parts: malevolence and competence.

Misconception #5 Avoiding Internet Explorer and Outlook makes you safe from ActiveX problems.
Many third-party applications take advantage of ActiveX controls as part of their ordinary operation. Additionally, third-party tools can execute scripts. Indeed, the Windows Scripting Host can be used by any application. Thus, avoiding Internet Explorer and Outlook does not guard you from all attacks based on ActiveX controls.

3. Active X Security Concerns and Risks

ActiveX is a powerful and, therefore, potentially dangerous technology. Most of the risks can be managed if you are aware of the problems. The concerns discussed in this section should be borne in mind when deciding when and how to use ActiveX in your environment.

Security concerns in ActiveX can be divided into two broad areas: 1) concerns related to importing and installing controls and 2) concerns related to running controls. This second set of concerns can be further divided into those directly related to controls and those related to the relationship between ActiveX and scripting.

Download Concerns—importing and installing controls

1. Ideally the decision to install software should be based on the capability of the software. Currently, there is no good way to do this with ActiveX controls. Instead, the decision must be based on the (presumed) source of the control. This is unsatisfactory for two reasons. First, the signer of the control may not be any more capable of assessing the control's safety than the end user is. Second, the end user must trust the distribution chain from the original source to the signer.

   The ultimate problem with any signature scheme is that safe controls can come from untrusted sources, and unsafe controls can come from trusted sources.

2. Signatures on ActiveX controls persist. Once an ActiveX control has been signed by a trustworthy party, the signed control is available to attackers. If a vulnerability is discovered in a signed ActiveX control, attackers can use the trust relationship between the victim and the signer to introduce the vulnerable control. Thus, a signed vulnerable control provides attackers a means to install remotely accessible software with known vulnerabilities onto a system if they can convince the victim to trust a "trustworthy" party.

3. In the current Windows architecture, a control need only be registered once per machine. This could lead to problems if a machine is shared by multiple users. Any one user can download a control, at which point it is available to all users on the machine.

4. Remediation of ActiveX controls is not always possible. Sometimes, it is not possible to implement desired functionality securely. See for example, http://www.cert.org/advisories/CA-2000-15.html and XEnroll (Appendix B, VU#3062; MS article Q242366, http://support.microsoft.com/support/kb/articles/Q242/3/66.asp).

Execution Concerns—running controls

1. ActiveX controls have more capabilities than tools that run strictly in a sandbox. Because ActiveX controls are native code that run directly on a physical machine, they are capable of accessing services and resources that are not available to code that runs in a restricted environment.

2. Nearly all ActiveX control security mechanisms are based on Internet Explorer. Unfortunately, ActiveX controls do not rely only on Internet Explorer; they can be installed and executed completely outside of IE. Third-party applications that use ActiveX technology may not provide the security mechanisms available in Internet Explorer.

3. Many of the security mechanisms provided by Internet Explorer are coarse. Some of the security mechanisms are all-or-nothing propositions, forcing a user to choose between functionality and security. For instance, there is currently no way to run a single “unsafe for scripting” control without enabling all “unsafe for scripting” controls.

4. When an ActiveX control is executed, it usually executes with the privileges of the current user. There is no mechanism for externally restricting the privileges of a control. (i.e., “sandboxing”).

5. Because ActiveX controls can be invoked remotely (through a web page or email message), each control presents a channel into a network that can potentially be used by an attacker.

6. ActiveX controls do not have an effective abstraction. Because each ActiveX control has arbitrary latitude in deciding when it can be run and what it can do, it is impossible for users to intuitively determine the behavior of an given control.

7. ActiveX controls are difficult to manage and audit, particularly for non-expert users. The tools to manage ActiveX controls are lacking in several important areas. (For more details, see the section for system and network administrators in Part 2.)

Scripting Concerns

1. ActiveX controls that are scriptable are responsible for implementing their own security. Because ActiveX controls do not run in a restricted environment (a sandbox), each scriptable control is required to implement its own security policy—in effect, its own sandbox. Implementing a single sandbox is difficult, as evidenced by recent problems in certain Java classes from Netscape (http://www.cert.org/advisories/CA-2000-15.html). Though it is not true that each control has to implement a general-purpose sandbox, each control does have to ensure that it behaves well in response to any input, in any order. Because each control presents a channel into your network (as described above), there is a large number of potential failure points.

2. Scripts can use controls in ways unanticipated by the original control author. This often leads to unexpected behavior that can be exploited to violate security policy.

3. Scripts can invoke controls “translucently.” A user may not realize that a script is using controls, and it may be impossible to determine beforehand which controls are being used (in an HTML document). As a result, the user might not be able to make informed decisions on whether to open such documents.

4. ActiveX is a means for scripts on a web page to escape the Internet Explorer “sandbox.” This is arguably a script engine issue, but it does illustrate one way in which a control could be used to subvert security.

5. Scripting engines other than those in Internet Explorer might not provide the security mechanisms that IE does with respect to ActiveX. As third-party applications incorporate ActiveX technology, they will be responsible for invoking ActiveX security.

6. Cross-site scripting is still poorly understood. Users of many web sites are vulnerable to a variety of cross-site scripting attacks. (See http://www.cert.org/advisories/CA-2000-02.html). Because ActiveX controls are not isolated in any way, trust decisions made in the context of a cross-site scripting attack can lead to the invocation of vulnerable native code.

4. ActiveX Benefits and Security Features

The use of ActiveX controls presents benefits as well as risks. Security features on your system can help you manage ActiveX controls. In this section, we list some key benefits and briefly describe the security features so that readers are familiar with the range of possibilities. Specific information about how to use the features can be found in Part 2 of this paper.

Benefits of ActiveX

These are among the primary benefits of using ActiveX controls:

ActiveX controls promote reuse. Because the invocation interface between ActiveX controls is standardized, system and application developers can reuse controls easily for a variety of purposes. This is true regardless of the choice of development tools or language.

ActiveX controls have more capabilities than tools that run strictly in a sandbox. Because ActiveX controls are native code that runs directly on a physical machine, they are capable of accessing services and resources that are not available to code that runs in a restricted environment. This presents the ability to do things that could not otherwise be easily accomplished. Several competing technologies, such as Java, offer similar capabilities.

ActiveX controls are available to meet a wide variety of needs. ActiveX controls are widely available for a variety of purposes. For many organizations, they represent the most convenient or cheapest path to gain desired or required functionality.

ActiveX Security Features

A variety of security features can be used to improve the security of working with ActiveX controls. They are listed below, and details about how to use them are in Part 2.

"Administrator Approved" setting – Within each Internet Explorer security zone (see below), there is an option to run only the controls that have been approved by

the administrator. The Administrator Approved setting can enforce policy even after an ActiveX control has been installed.

Authenticode – This family of technologies is used to digitally sign and verify executable content, and to control the download of code to the workstation.

CodeBaseSearchPath – With the CodeBaseSearchPath registry key, administrators can control where the system will look when it attempts to download and install ActiveX controls.

Internet Explorer Administration Kit (IEAK) – The IEAK is a tool that can be used by administrators or ISPs for tasks such as centrally controlling and creating Internet Explorer settings, distributing them to users, and centrally managing Authenticode settings.

IObjectSafety – The IObjectSafety interface provides a method, besides “Safe for” flags, for determining the operations for which an ActiveX control is safe.

“Safe for” flags – The “Safe for Initialization” and “Safe for Scripting” flags, usually set by an ActiveX control when it is installed, determine the action a web page can take with an ActiveX control.

Kill bit – The kill bit is a registry value that prevents Internet Explorer from loading an ActiveX control. It cannot be overridden by any security zone configurations.

Security zones – The security zones feature of Internet Explorer (and other products that rely on security zones) can be used to tightly control the behavior and accessibility of controls. Internet Explorer includes the ability to group web sites into four zones plus a fifth built-in zone, called the “My Computer” zone, which must be configured with the Internet Explorer Administration Kit and cannot be configured from the browser.

Windows 2000 Group Policy Objects (GPO) – Like the IEAK, the Group Policy Objects can be used to define Internet Explorer configurations and manage Authenticode settings centrally.

PART 2 : Suggestions for Specific Groups

Part 2 of this report builds on the general information in Part 1 and provides information suited to the needs of various groups involved with ActiveX: managers, system administrators and security personnel, developers of ActiveX controls, and individuals who administer their own computers.

5. Managers

We encourage managers to take the steps listed below. Because managers' responsibilities do not change extensively from one security problem to another, much of this advice may be familiar to managers who are already actively involved in security issues. They are, nevertheless, important activities in managing security risk.

- Become fully informed with regard to the nature of ActiveX security issues and the potential ramifications of decisions related to their use. Stay informed about the security issues as technology changes and new threats arise. Senior management should receive direct briefings from security staff in an effort to facilitate understanding.

- Be cognizant of your own site's security posture.

- Develop an augmentation strategy to provide staff and other resources in the event of a computer security compromise. Determine which staff may be needed and where they should report. Be sure there are phone or alternative communications in case electronic communication becomes difficult or impossible.

- Be sure your staff has the time and resources needed to perform the activities described under "System Administrators" below.

- Ensure privacy issues associated with log retention and review have been addressed in policy and that adequate analytical information is readily available to critical staff in the event an attack occurs.

- Examine your current policy requirements. In particular, ensure responsibility is defined for 1) enforcing minimum security standards; 2) cutting off users (even executive-level users) whose accounts may have been compromised or are at risk; and 3) disconnecting uncontrolled Internet connections.

- Be sure that all levels of management understand and are held accountable for security planning and implementation. Be sure that an adequate and enforceable acceptable-use policy exists enterprise wide.

- Realize that the escalating Internet threat environment must be matched by corresponding investments in security. Define security resources in the budget.

- Examine your current network and security architecture. Many sites have optimized connectivity for speed of access, making decisions that complicate security measures. In the escalating threat environment, speed and reliability can be denied unless security is included in the architecture.

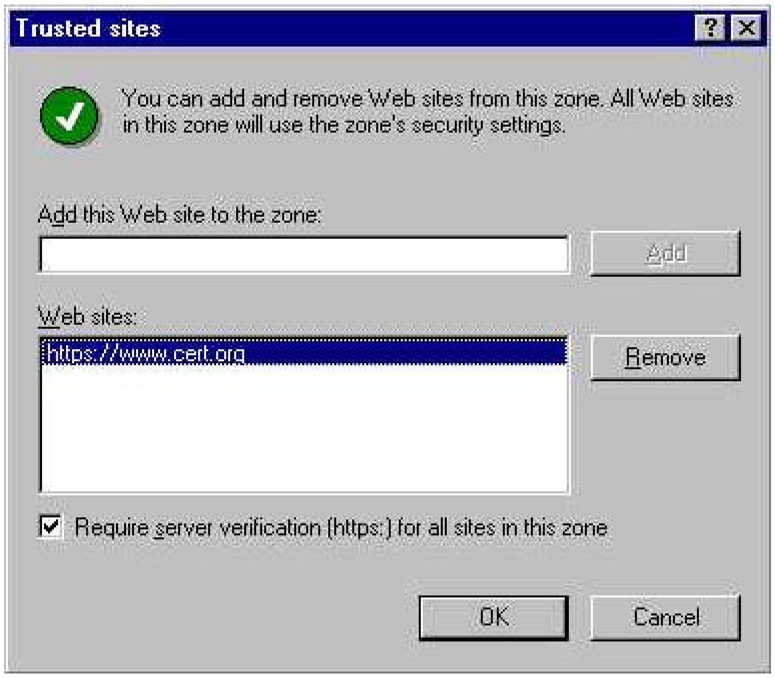
6. System Administrators and Security Personnel

This section begins with a possible configuration, identified during the Security in ActiveX Workshop, that enables you to improve the security of using ActiveX controls. This section is followed by detailed information on tools and procedures using the security features listed in Part 1, Section 4, “ActiveX Benefits and Security Features.”

One Possible Configuration

Security risks in ActiveX controls can be mitigated by conservative configuration choices. Below is one set of configuration choices:

- Use highly restricted zones. The security zones feature of Internet Explorer (and other products that rely on security zones) can be used to tightly control the behavior and accessibility of controls depending on the zone of origination of the request for the control.
- Secure the Trusted zone with SSL, as shown below. The Trusted zone should be configured to require SSL authentication to prevent attacks against the DNS system from affecting the security of the local system.


Trusted sites
?
You can add and remove Web sites from this zone. All Web sites in this zone will use the zone's security settings.
Add this Web site to the zone:
Add
Web sites:
Remove
OK
Cancel

https://www.cert.org

[✓] Require server verification (https:) for all sites in this zone

Attacks against DNS systems are among the more common attacks reported to the CERT Coordination Center. For example, see http://www.cert.org/advisories/CA-2000-03.html. Consider taking advantage of these security features, consulting the Procedures section for details on how to use them:

- Set CodeBaseSearchPath. The use of the CodeBaseSearchPath registry key can control where your system will look when it attempts to download ActiveX controls.

- Use the Internet Explorer Administration Kit (IEAK) to define and manage the set of runnable controls. The IEAK can be used to define and dynamically manage the set of controls that are scriptable.

- Modify the UserAgent tag through the IEAK. The IEAK can also be used to modify the UserAgent tag of produced by Internet Explorer. In conjunction with a firewall that blocks outbound HTTP connections based on the UserAgent tag, this can reduce the chance that versions of Internet Explorer not managed by the IEAK can reach external (and possibly malicious) sites (provided that internal users are not intentionally trying to subvert the policy).

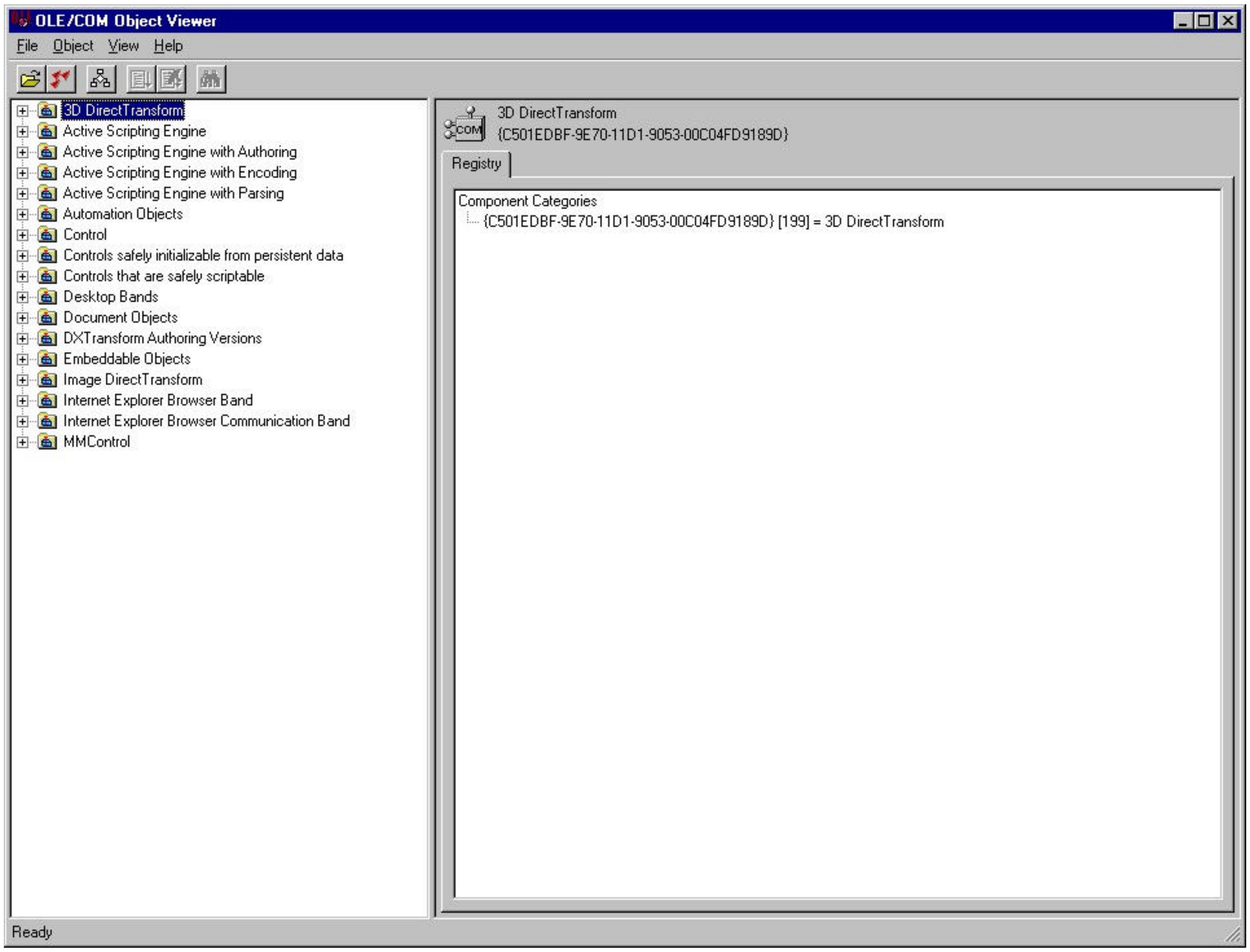
Auditing Tools, Monitors, and Filters

ActiveX is a complex technology with a great many options. In addition, there are thousands of different controls, any one of which may present a vulnerability. Managing the complexity and the sheer number of ActiveX controls is perhaps the most difficult task in dealing with ActiveX controls from a security perspective.

Auditing Tools

A substantial problem when using ActiveX controls is the lack of advanced auditing tools. Indeed, it may be impossible to audit certain security-related aspects of a given ActiveX control at all, namely, the way in which a given control will respond to when queried on its IObjectSafety interface. This interface determines whether a control is safe to run in the given context, and the decision parameters are entirely up to the developer of the control. For example, the developer could decide that a control will run if activated on Tuesdays, if a certain file exists, or if an arbitrary condition of the caller is met. Because ActiveX controls can be invoked remotely (through a web page or email message), each control presents a channel into a network that can potentially be used by an attacker. The inability for a network manager or system administrator to determine the circumstances under which these channels can be used presents a substantial challenge to securing networks.

That said, the most common auditing tool for examining ActiveX controls is Microsoft's OLEVIEW. OLEVIEW lists a variety of information about COM objects installed on a system (see the example below). Unfortunately, OLEVIEW does not give administrators the ability to determine if a control is scriptable. The "Safe for Scripting" and "Safe for Initialization" settings do not tell the whole story. In fact, Microsoft recommends that developers implement the IObjectSafety interface. The behavior of controls that implement IObjectSafety is opaque.



3D DirectTransform
Active Scripting Engine
Active Scripting Engine with Authoring
Active Scripting Engine with Encoding
Active Scripting Engine with Parsing
Automation Objects
Control
Controls safely initializable from persistent data
Controls that are safely scriptable
Desktop Bands
Document Objects
DXTransform Authoring Versions
Embeddable Objects
Image DirectTransform
Internet Explorer Browser Band
Internet Explorer Browser Communication Band
MMControl

3D DirectTransform
{C501EDBF-9E70-11D1-9053-00C04FD9189D}

Component Categories
{C501EDBF-9E70-11D1-9053-00C04FD9189D} [199] = 3D DirectTransform

It is possible to write tools that emulate common scenarios and to methodically test controls for scriptability in each of the common scenarios. Doing this would provide a substantially greater degree of auditability but cannot reveal the entire picture, particularly if there is a “benevolent” back door in a control or if there is a set of unusual circumstances that would cause the control to be scriptable. A market opportunity may exist for such a tool.

Monitors and Filters

Some firewalls can monitor and selectively filter the invocation and downloading of ActiveX controls. Firewalls that do not also act as a key proxy for SSL-encrypted sessions may be easily defeated by an attacker’s establishing an SSL-secured session with the victim. If a site relies exclusively on a firewall to prevent the introduction or invocation of vulnerable or malicious ActiveX controls, and that firewall does not buffer SSL-sessions, then the site is at risk. As with any existing security technology, a firewall is not a magic bullet.

Procedures for Using ActiveX Security Features

This section provides detailed instructions on how to use the security features described in Part 1 of this report.

Security Zones

Within the Internet Explorer browser, there are many security settings that administrators or users can configure to properly manage the risk from mobile code. Despite having all these options, it is often impossible to define a single security policy that is applicable to all sites. Consider the corporate intranet, where web sites may use rich content such as ActiveX controls, unrestricted Java applets, and extensive scripting for tasks such as entering payroll data or performing other employee self-service actions. At the other end of the spectrum are sites that you know to be of questionable intent; however, users may still need to visit these sites to gather information. You want to permit this with adequate safety measures. The bottom line is that all web sites are not trusted equally, and security policy must be flexible enough to address this.

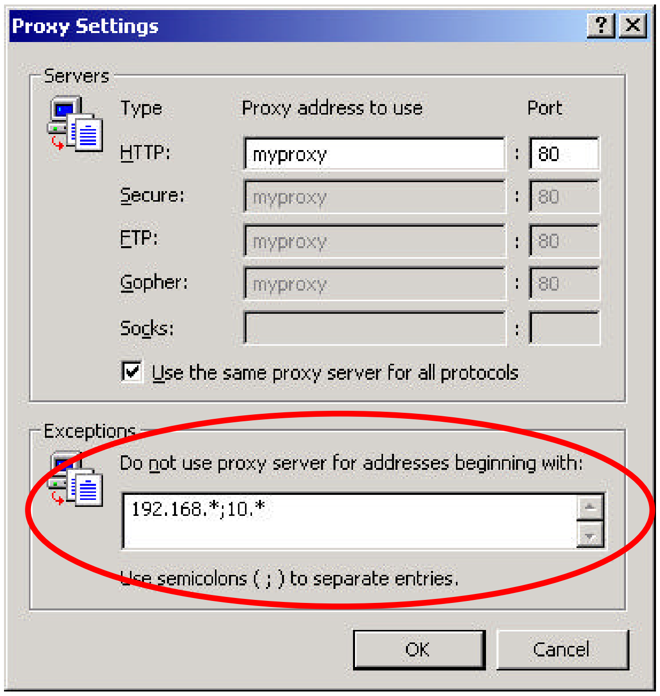
To address this issue, Internet Explorer includes the ability to group web sites into “security zones.” Each zone can then have the appropriate security policy applied to it.

Internet Explorer 4.0 and higher are pre-configured with four security zones, and additional zones can be added programmatically using Internet Explorer Zone Manager API. The four default zones are described as follows:

1. Local Intranet zone: Sites are considered a part of this zone by the following three sets of criteria, each of which may be individually disabled.

    a. The site is considered on the local intranet and is not listed in any other zone. When a Microsoft Proxy server is used to separate your intranet from the Internet, then the list of local intranet sites is downloaded to the browser from the proxy's Local Address Table (LAT).

    b. The site is defined in the Internet Explorer proxy settings as an address to bypass the proxy server to access.


Proxy Settings

Servers

Type	Proxy address to use	Port
HTTP:	myproxy	80
Secure:	myproxy	80
FTP:	myproxy	80
Gopher:	myproxy	80
Socks:		

[✓] Use the same proxy server for all protocols

Exceptions

Do not use proxy server for addresses beginning with:

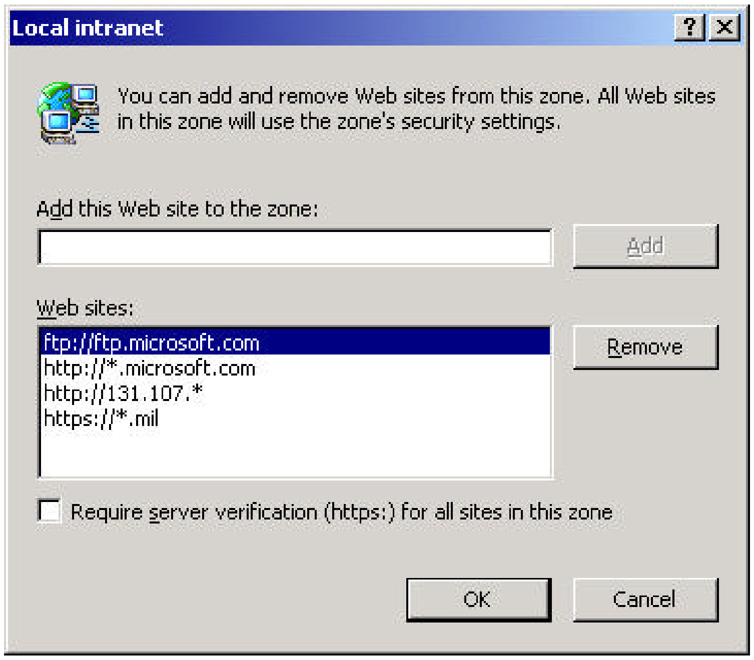
192.168.*;10.*

Use semicolons (;) to separate entries.

c. The address being referenced is a UNC path (e.g., \\server\share\folder\file.htm) or the server name does not have a "." in it.

In addition to these criteria, you can add sites directly to the Local Intranet zone by host name, IP address, DNS domain, or IP subnet. In IE 5.0 or later, the sites added directly to the zone can also be required to use SSL/TLS for server authentication in order to be considered in the Local Intranet zone.

The default security level for the Local Intranet zone is Medium.


Local intranet

You can add and remove Web sites from this zone. All Web sites in this zone will use the zone's security settings.

Add this Web site to the zone:

Web sites:

ftp://ftp.microsoft.com
http://*.microsoft.com
http://131.107.*
https://*.mil

[ ] Require server verification (https:) for all sites in this zone

2. Trusted Sites zone: This zone contains sites you trust—sites that you believe you can download or run files from without worrying about damage to your computer or data. You can assign sites to this zone by their name, IP address, DNS domain, or IP subnet just as with the Local Intranet zone. In this zone, you can require that a site use SSL/TLS for server authentication in order to be considered in the Trusted Sites zone (IE 5.0 and later only). The default security level for the Trusted Sites zone is Low.

3. Restricted Sites zone: This zone contains sites you don't trust—that is, sites that you're not sure whether you can download or run files from without damage to your computer or data. You can assign sites to this zone using the same tools as with the Local Intranet and Trusted Sites zones; however, the option to require SSL/TLS is not available. The default security level for the Restricted Sites zone is High.

4. Internet zone: By default, this zone contains anything that is not on your computer or an intranet, or assigned to any other zone. The default security level for the Internet zone is Medium.

There is a fifth built-in zone called the My Computer zone (which contains files on your local computer) that is configurable only from the Microsoft IEAK. These settings are not available in the browser interface.

The pre-configured security levels of the zones should provide adequate security to most organizations when sites are categorized appropriately. Each of the 106 security settings within each zone can be customized to meet organizational needs.

These are some of the security settings defined within each zone:

- Download signed ActiveX controls (Disable/Enable/Prompt)
- Download unsigned ActiveX controls (Disable/Enable/Prompt)
- Initialize and script ActiveX controls not marked as safe (Disable/Enable/Prompt)
- Run ActiveX controls and plug-ins (Administrator approved/Disable/ Enable/Prompt)
- Allow cookies that are stored on your computer (Disable/Enable/Prompt)
- Allow per-session cookies (Disable/Enable/Prompt)
- Active scripting (Disable/Enable/Prompt)
- Scripting of Java applets (Disable/Enable/Prompt)
- A variety of Java VM security restrictions

It is easiest to customize these settings using the Internet Explorer Internet Options dialog. In addition, corporate administrators or ISPs can use the IEAK or Windows 2000 Group Policy Objects (GPO) to define the configuration centrally for their clients. The IEAK allows corporate administrators or ISPs to create a customized version of Internet Explorer that can then be distributed to their users on CDs or over the network. The customized version can come preset with any of the desired security settings discussed in this document.

However, if you do not have the ability to distribute these changes using the IEAK or GPO, you can find the zone security settings in the following registry key:

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
\Internet Settings\Zones]
```

Under this registry key is a series of subkeys that each store the settings for a particular zone:

0 = My Computer zone
1 = Local Intranet zone
2 = Trusted Sites zone
3 = Internet zone
4 = Restricted Sites zone

The definition of how sites are categorized into zones is stored beneath the registry key:

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
\Internet Settings\ZoneMap]
```

We recommend that you still set these keys using the user interface and then export the registry key(s) to a .REG file for installation on machines. Note that these settings are stored in the HKEY_CURRENT_USER hive of the registry and are, therefore, configurable on a per-user basis. However, identical registry keys exist in the HKEY_LOCAL_MACHINE registry hive that will place the same restrictions on all users of a particular machine. Command-line tools such as *regedit /s foo.reg* can be used as a part of a logon script or distributed via the Server Management System (SMS) to configure the zone settings for a user. Further details on the registry keys used by Internet Explorer security zones is available in Microsoft Knowledge Base article Q182569.

Microsoft Outlook (98 and higher) and Outlook Express (4.0 and higher) can also take advantage of security zones. In the Outlook security configuration, you have the option of determining which zone security settings will be used when viewing HTML email. Note that you can only choose from the Internet zone or the Restricted Sites zone, and modifications to the zone settings affect Internet Explorer as well as Outlook and Outlook Express.

For further information on how to programmatically configure the Internet Explorer zones, the effect of specific zone security settings, or how to create new zones, please consult the MSDN online library at http://msdn.microsoft.com/workshop/security/szone/urlzones.asp

Authenticode

Microsoft Authenticode is the name used to describe a family of technologies for digitally signing and verifying executable content. A person developing an ActiveX control or Java applet can digitally sign the software cabinet (.CAB) file using the private key from a digital certificate. This certificate can be obtained from a commercial Certificate

Authority, such as Verisign, GTE or Thawte, or from a corporate Certificate Authority such as the Windows 2000 Certificate Services.

The Microsoft code-signing tool *signcode.exe* is used to sign the program; it prompts the signer for a certificate stored in the Microsoft Crypto API. Certificates and their associated private keys are automatically stored in the Microsoft Crypto API if they are requested from a web page using Internet Explorer. If the certificate is obtained by other means, it can be loaded into the Crypto API using the PKCS#12 (.p12 or .pfx file) format and the certificate import wizard.

The *signcode.exe* tool signs files only if the certificate used asserts the value for Code Signing (1.3.6.1.5.5.7.3.3) in the Enhanced Key Usage certificate extension. The code-signing tool can be downloaded from http://msdn.microsoft.com/downloads/ in the Security \ Authenticode section.

Internet Explorer automatically validates the digital signature on the mobile code when the user downloads the CAB file. A valid signature tells the user who created the code and verifies that the code has not been altered since the author signed it. This allows the person downloading the code to make a trust decision about the author of the program: "Do I believe that this person/company produces software that is free of viruses and would not attempt any malicious actions?" If you consider the traditional method of distributing software by walking into a computer store and purchasing it off the shelf, the Authenticode signature acts like the shrink-wrapped software box and "Certificate of Authenticity." The fact that the software is signed does not attest to the quality any more than the fact the software is in a shrink-wrapped box does.

Not all users want to make a trust decision every time they install a new piece of software, and in a corporate environment it may be advisable to have a central authority establish rules that guide which software is installed. The Internet Explorer Administration Kit allows corporate administrators to customize the Authenticode policy settings and develop a custom browser configuration that can be deployed to users. This custom configuration could prevent users from running any unsigned code or allow them to only run code signed by specific authors. For example, you might establish a process that calls for all internally developed code to be approved before deployment—it could be a part of your existing configuration management process. This process might even include a code review to ensure that the program can't be used for malicious actions. Once this program has been approved for deployment, the configuration control body would digitally sign it. To enforce this process, all browsers within the company would set their Authenticode settings to run only controls signed by this configuration control board's certificate.

Furthermore, these Authenticode settings can be locked down and centrally managed using the IEAK. For users running in Windows 2000 domains, the management capabilities of the IEAK are also available as Group Policy Objects that can be applied to domains or organizational units.

Since Java applets are run within a virtual machine (VM) in the browser, IE coupled with Authenticode technology can be used to define the level of trust required to run an applet. An author signing a Java applet can state in the signature which permissions the applet requires. For example, the applet might need to access the local hard disk or transmit data across the network. The user or network administrator can define which rights applets should be permitted, and the Microsoft Java VM will check that the applet can run within the specified policy prior to loading it. Should a running applet attempt an action that it did not request in its Authenticode signature, the VM will stop execution of the applet. The list of available permissions with which a Java applet can be signed is available at http://www.microsoft.com/java/sdk/40/start.htm?http://www.microsoft.com/java/sdk/40/pg/pg_pkgdist_sgncd.htm

CodeBaseSearchPath

Most web pages that use ActiveX controls also provide the client with information on where to download and install the control. This information is provided using the CODEBASE keyword in the HTML tag and references the URL of the component itself or a CAB file containing the control and installation instructions (INF file).

The following HTML code snippet is from the investor.msn.com home page:

```
<OBJECT type="application/x-oleobject"
classid="clsid:62360003-D8A7-418b-9DC6-2B9DE95273A0"
Codebase="http://fdl.msn.com/public/investor/v8/0326/ticker.
cab#version=8,2000,0326,2" width=100% height=34>
</OBJECT>
```

The OBJECT tag is used to load an ActiveX control within a web page. When Internet Explorer parses this HTML, it will first check to see if the object is already installed by searching for the classid listed in bold above. If the object is installed, then the registry entry lists the program file to run. If the classid cannot be found in the registry, then by default IE looks for the CODEBASE keyword in the HTML tag. In the example above, Internet Explorer is given a hint by the web page author that the desired control can be found as ticker.cab on the web site http://fdl.msn.com.

However, if IE cannot access this location or if the CODEBASE tag is not present, there is a list of alternate locations stored in the following registry value:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
\Internet Settings\CodeBaseSearchPath
```

The default data in this registry value is

```
CODEBASE;<http://activex.microsoft.com/objects/ocget.dll>;<h
ttp://codecs.microsoft.com/isapi/ocget.dll>
```

The first entry tells IE to always use the URL provided in the CODEBASE HTML tag first to download the code. If that fails, IE will step through the list of alternate sites listed in the CodeBaseSearchPath until it finds the code or runs out of places to check.

For organizations that want to permit the use of ActiveX controls but do not want to allow users to download code directly from the Internet, this CodeBaseSearchPath can be customized to something like the following:

```
<http://your_internal_site/get.asp>
```

Removing the initial CODEBASE keyword prevents users from downloading any code from the Internet, while substituting the other locations with an internal site allows the administrators to specify which controls are downloaded. The URL specified in the CodeBaseSearchPath will receive an HTTP POST request with data in the following format and respond with the object to install and load.

```
CLSID={class id}
Version=a,b,c,d
MIMETYPE=mimetype
```

Additional information on how this request should be processed is available at http://msdn.microsoft.com/workshop/delivery/download/overview/implementation.asp.

The sites that receive and process requests for objects are called Object Stores. When implemented on top of an IIS web server, these stores could also be configured to authenticate the user prior to allowing the download of the code. For instance, specific users within your enterprise could be permitted to download controls to their machines that you may not want all users to access.

Once a control is downloaded and installed on the system, it is then available to all users that log on to the system. However, Internet Explorer provides other means to control objects after they have been installed.

The "Safe for" Flags and IObjectSafety

Once a control is installed on a system, any HTML web page can attempt to load it using the OBJECT tag. However, many applications that the user may install are not intended to be loaded or scripted within web pages, and doing so would allow a malicious web site to damage the user's machine. To prevent this, there are two flags that can be set on each COM object registered on the workstation that determine the action a web page can take with this control. These flags are referred to as "Safe for Initialization" and "Safe for Scripting" and are usually set by the control when it is installed.

The following quotes from the MSDN Web Workshop describe the assertion made by setting these flags:

> "If you mark your control as safe for initializing, you are asserting
> that no matter what values are used to initialize your control, it

> won't do anything that would damage a user's system or
> compromise the user's security.
>
> "If you mark your control as safe for scripting, you are asserting that your control won't do anything to damage a user's system or compromise the user's security regardless of how your control's methods and properties are manipulated by the Web page's script. In other words, it has to accept any method calls (with any parameters) and/or property manipulations in any order without doing anything bad."

It is not recommended that you change these settings to mark a control safe that the author had not marked as safe. However, these flags may be used to further restrict controls that you believe pose a security risk to your users. The flags can be set directly in the registry by creating the following keys:

```
[HKEY_CLASSES_ROOT\CLSID\{CLSID of control}\Implemented
Categories\{7DD95801-9882-11CF-9FA9-00AA006C42C4}]
```

(Marks a control as "Safe for Scripting")

```
[HKEY_CLASSES_ROOT\CLSID\{CLSID of control}\Implemented
Categories\{7DD95802-9882-11CF-9FA9-00AA006C42C4}]
```

(Marks a control as "Safe for Initialization")

If you remove the appropriate bolded key above, a control will no longer be marked as safe for the operation.

Another method for determining the operations a control is safe for is to query the IObjectSafety interface, if the control supports it. A control can chose to implement IObjectSafety in addition to or instead of the registry keys above. Prior to loading the control and prior to checking the registry keys, Internet Explorer queries the IObjectSafety interface on the object to see if the control supports it. If the control supports this interface, then IE will use the interface to query for safe operations of the control instead of using the registry keys. The control may implement the IObjectSafety interface by simply querying the above registry keys or it may tailor its security depending on the zone it is in or the current IE security policy settings.

The Internet Explorer security settings also allow the user or administrator to override the behavior of the "Safe for" flags depending on the security zone in the following manner:

- Permit the initialization and scripting of controls even though they are not marked as safe for such operations.
- Disable the scripting of controls even though they are marked as safe for scripting.

For instance, you might decide to permit the initialization and scripting of a control not marked as safe in the Trusted Sites zone because you know the web authors of those sites would not attempt any malicious actions on your workstation. However, in the Restricted

Sites zone, you might decide to prevent the scripting of controls even though they are marked safe. By using these measures instead of directly manipulating the “Safe for” flags, you can continue to rely on the “Safe for” flags as configured by the component author in the Internet and Local Intranet zones.

“Administrator Approved”

Within each IE security zone there is an option to run only the controls that have been administrator approved. While settings such as Authenticode are used to control the download of code to the workstation, the “Administrator Approved” setting can enforce security policy after a control has already been installed.

The list of controls that are permitted to run are stored in the registry in the following value:

```
[HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Curre
ntVersion\Internet Settings\AllowedControls]
"{8856F961-340A-11D0-A96B-00C04FD705A2}"=dword:00000000
```

The bolded text in quotes is the CLSID of the object that you wish to allow access. Once the “only run administrator approved controls” option is set, only those values with a DWORD value of 00000000 will run. Note that this setting is stored in the HKEY_CURRENT_USER hive of the registry and is, therefore, configurable on a per-user basis. Command-line tools such as *regedit /s foo.reg* can be used as a part of a logon script or distributed via SMS to configure the permitted controls for a user.

On Windows NT and Windows 2000 machines, this registry key normally has an access control list (ACL) that prevents users from changing the list of approved controls. However, the Windows 9x platform does not support registry ACLs; therefore, a user could change the approved control list by directly manipulating the registry. You can use the Windows 9x system policy mechanisms to reset the registry values at logon and prevent the use of the registry editor tool.

Though it is possible to set the list of approved controls directly via the registry, it is far easier to use the Internet Explorer Administration Kit. When the IEAK is installed on the administrator’s machine, the list of controls that can be marked as approved is created from the controls already present on the machine. To manually change this list, edit the *axaa.inf* file in the IEAK installation directory.

On Windows 2000 machines in a domain, the configuration of “Administrator Approved” controls can be distributed to users using Group Policy Objects in the Active Directory. Like any Group Policy settings, these can be applied domain wide, site wide, or to users in specific organizational units.

The “Kill Bit”

On occasion, a control is installed that poses such a risk that it should not be accessible from within the browser under any circumstances. Simply marking the control as “unsafe for initialization” is not sufficient to ensure that the control will never load, even after a re-install scenario. This is where the kill bit comes in, and it cannot be overridden with any of Internet Explorer’s zone configurations.

The kill bit is a registry value that contains the CLSID of all controls that IE will not load. To set the kill bit on a specific control, you must create or locate the following registry key:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet
Explorer\ActiveX Compatibility\{8856F961-340A-11D0-A96B-
00C04FD705A2}]

"Compatibility Flag"=dword:00000400
```

The bolded text in quotes is the CLSID of the object that you are setting the kill bit for, and the DWORD value of 00000400 prevents IE from loading the control.

7. Developers of ActiveX Controls

This section provides guidelines and pointers to information that can help developers of ActiveX controls mitigate the risks inherent in implementing mobile-code technology, with a focus on ActiveX code in particular.

First, developers who create ActiveX controls need to follow the same common-sense guidelines that apply to development within any complex software environment:

- Consider risks and benefits of using mobile-code technology to fulfill design requirements.
- Follow good software engineering practices.
- Pay special attention to avoiding known software defects that may lead to security vulnerabilities and exposures.

In addition, there are several additional steps needed to safely integrate software components into the Microsoft ActiveX framework:

- Follow Microsoft’s recommendations for creating ActiveX controls.
- Stay abreast of current trends in ActiveX security vulnerabilities to ensure no gaps in past knowledge lead to future compromises.
- Watch for new changes in ActiveX development in the evolving .NET framework.

The following subsections elaborate on these points further.

Consider risks and benefits of using mobile-code technology to fulfill design requirements.

This is simply a reminder that when you consider using mobile-code technology, it is helpful to examine the alternatives and determine the safest way to meet the design requirements.

Follow good software engineering practices.
As always, good software engineering practices help developers avoid pitfalls, security and otherwise. Here are a few examples of good practice.

Assume the least functionality necessary to perform a required task.
ActiveX controls should run with the minimum functionality required to accomplish the tasks they need to perform, and no others. Each control should be designed to make only the calls to the system needed to fulfill functional specifications. For example, if an ActiveX control is designed to monitor a system and display its output to a screen, it does not need to log activity to a file and it should not be written to access routines for writing to files.

Document your work.
Ensure that you thoroughly document the behavior of the control you develop so system administrators and users know what to expect and can take steps they consider appropriate for running your control.

Avoid using insecurely managed temporary space; use the Internet Download Cache.
There is a discussion of the Temporary Internet Files folder at
http://msdn.microsoft.com/library/psdk/shellcc/shell/Functions/CSIDL.htm

This web page describes the CSIDL_INTERNET_CACHE Version 4.72 as the "File system directory that serves as a common repository for temporary Internet files. A typical path is C:\Documents and Settings\username\Temporary Internet Files."

Be aware of the potential for abuse by malicious data.
Where possible, avoid executing data used by a control. Controls that can treat data inputs as instructions are open to attacks. These attacks leverage information used as input by a control that are actually instructions to be executed by the system without the knowledge or approval of the user. Further discussion of the general concept can be found at
http://www.fish.com/security/maldata.html and
http://csrc.nist.gov/nissc/1996/papers/NISSC96/paper048/MALDATA.PDF.

One example of the way an attacker could violate trust relationships inherent in the ActiveX security model is known as cross-site scripting. For more information see
http://www.cert.org/advisories/CA-2000-02.html

Pay special attention to avoiding other known software defects such as
- buffer overruns
- input validation errors (printf format strings checking)
- symbolic link ambiguities
- race conditions
- trusting untrustworthy information

There are many examples of controls that do not appear to have been crafted using such practices. Some specific examples of buggy controls audited by Microsoft are listed in Microsoft's Knowledge Base article Q170770:
http://support.microsoft.com/support/kb/articles/Q170/7/70.ASP

Other examples are listed in Appendix B and in the References listed at the end of this paper.

Follow Microsoft recommendations for creating ActiveX controls.
A good source of general Microsoft recommendations for developers can be found on the Microsoft web site for security-minded developers at
http://www.microsoft.com/security/devlink.asp

Specific advice for ActiveX developers is also available. Although the following documents seem dated in some respects, they still contain much good advice for the ActiveX programmer. Some excerpts and comments follow:

Make sure your control is safe before asserting it is.
It is the responsibility of the ActiveX control developer to place reasonable limits on the controls running in a user's environment. For example, controls that allow remote commands to run, delete arbitrary files, or modify registry entries should generally not be able to be invoked via scripts.

Other guidelines from Microsoft on this issue can be found in "Designing Safe ActiveX Controls" located at
http://msdn.microsoft.com/workshop/components/activex/security.asp.
This document contains Microsoft's ActiveX Safety Checklist:

> "Safety of the control is ultimately a subjective judgment, but as a general rule, *safe* means that none of the following undesirable effects will result from any conceivable use of the control:
>
> - Accessing information about the local computer or user.
> - Exposing private information on the local computer or network.
> - Modifying or destroying information on the local computer or network.
> - Faulting of the control and potentially crashing the browser.
> - Consuming excessive time or resources such as memory.
> - Executing potentially damaging system calls, including executing files.
> - Using the control in a deceptive manner and causing unexpected results."

Learn the IObjectSafety Interface and use it.
The IObjectSafety interface provides a finer granularity of control than setting registry bits when an administrator or developer tries to determine the safety status of ActiveX controls. It is best to assume that the environment in which a control is instantiated may

be one the developer did not intend the control to run in. IObjectSafety gives a developer more flexibility in trying to verify that if the control asserts it is safe, it will be much less likely to be intentionally used to cause damage to an end user's system.

IObjectSafety, when used in conjunction with the URL Security Zones API, should be considered when such flexibility is needed. However, registry bits can still be used to flag when controls may not be safe for use or scripting. These Microsoft documents discuss IObjectSafety, URL Security Zones, and registry settings in more detail:

- "Safe Initialization and Scripting for ActiveX Controls," at http://msdn.microsoft.com/workshop/components/activex/safety.asp, contains information regarding the use of "Safe for Scripting" and "Safe for Initialization" settings, as well as IObjectSafety:

  > "The **IObjectSafety** interface allows a container to ask a control to make itself safe, or to retrieve the current initialization or scripting capabilities for the control. This interface is defined in the Objsafe.h file. Currently two capabilities are supported: safe for scripting and safe for initialization."

- "IObject Safety Extensions for Internet Explorer," at http://msdn.microsoft.com/workshop/components/com/IObjectSafetyExtensions.asp, contains information about extensions to the IObjectSafety model and samples of Object Creation code:

  > "Controls need to implement these extensions only if they want to fully support Internet Explorer's security model. ... When the INTERFACE_USES_SECURITY_MANAGER bit is enabled on an object, the object must use the Security Manager provided by Internet Explorer 4.0 to make security decisions. Currently, the only scenario where the security manager needs to be queried is when objects are created."

- "URL Security Zones API Overview," at http://msdn.microsoft.com/workshop/security/szone/overview/overview.asp, describes Microsoft's URL Security Zones interfaces:

  > "Applications can manage the default URL security zone settings by using the IInternetZoneManager interface. Any change made using IInternetZoneManager will not be static, because the user could override them. In most cases, applications that need to control the URL security zone settings should create an application that hosts the WebBrowser control or MSHTML and implement their own security manager."

Use Microsoft's Authenticode code-signing tools
You can get more information about Authenticode and signing code with Microsoft Authenticode technology from http://www.microsoft.com/technet/security/authtech.asp

"Software Publisher Digital IDs^{SM} for Microsoft Authenticode^{TM} Technology," can be found at
http://www.verisign.com/developer/rsc/gd/authenticode/
http://www.verisign.com/rsc/gd/dev/authenticode/intro.html

Some Examples
Here is a dated example of creating a safe control from scratch:
http://support.microsoft.com/support/kb/articles/Q164/1/19.asp

Stay abreast of current trends in security vulnerabilities.
New vulnerabilities are uncovered all the time, so it is important to keep current to ensure no gaps in past knowledge lead to future compromises. Sources of more information about security vulnerabilities, including those in ActiveX technologies or implementations include

Microsoft Corporation (http://www.microsoft.com/security/)
CERT/CC Vulnerability Notes Knowledgebase (http://kb.cert.org/vuls)
CVE: Common Vulnerabilities and Exposures (http://www.cve.mitre.org/)

Watch for new changes in ActiveX development in the evolving .NET framework.
Microsoft has alluded to significant changes in the approach used to build ActiveX controls in the future using C-Sharp (C#) and other languages that can take advantage of their .NET framework. Here is an excerpt from
http://msdn.microsoft.com/vstudio/nextgen/technology/csharpintro.asp

> "With C#, every object is automatically a COM object. Developers no longer have to explicitly implement IUnknown and other COM interfaces. Instead, those features are built in. Similarly, C# programs can natively use existing COM objects, no matter what language was used to author them."

This isn't just a feature of the C# environment. The idea that all objects created under the .NET framework are "COM" objects is part of the Microsoft "Common Language Runtime," environment, a core concept in the .NET architecture. Part of this unifying concept is explained in http://www.microsoft.com/net/developer/framework_com.asp as follows:

> "Despite its name, the Common Language Runtime actually has a role in a component's development time and run-time experiences. While the component is running, the runtime is responsible for managing memory allocation, starting up and killing threads and processes, enforcing security policy, as well as satisfying any dependencies that the component may have on other components.
> ...

> *“Relationship to COM*
> “One of the primary goals of the .NET Framework is to make COM development easier. One of the hardest things about COM development is simply dealing with the COM infrastructure. Consequently, to make COM development easier, the .NET Framework automates virtually all of what developers currently think of as “COM”, including reference-counting, interface description, and registration.
>
> “It’s important to note that this doesn’t mean that .NET Framework components aren’t COM components. In fact, a COM developer using Visual Studio 6.0 could call a .NET Framework component and, to the developer, it would look like a COM component, complete with iUnknown data. Conversely, a .NET Framework developer using Visual Studio.NET would see a COM component as a .NET Framework component.
>
> “There is a caveat to this relationship: COM developers must manually do many of the things that .NET Framework developers can rely on the runtime to automate for them. For example, the security of a COM component must be written manually, and its memory can’t be automatically managed, and, to install a COM component, entries must be placed in the Windows registry. For .NET Framework components, the runtime automates these features. Components are self-describing, for example, and can therefore be installed without registering them in the Windows registry.”

In summary, developers who use the .NET framework with a tool like Visual Studio.NET have more ways to automate the specification and implementation of ActiveX/COM interfaces like IObjectSafety(). Conversely, this means the developer will bear a higher responsibility for exercising and testing the generated interface to ensure it behaves as desired and expected, and enforces an appropriate security policy, not just one assigned by default.

Additional Reading for Developers
Developers can find helpful information in these documents:

Component Development TOC
http://msdn.microsoft.com/workshop/components/toc.htm

Introduction to ActiveX Controls
http://msdn.microsoft.com/workshop/components/activex/intro.asp

ActiveX Control Tutorial
http://msdn.microsoft.com/workshop/components/activex/tutorial.asp

Building ActiveX Controls for Internet Explorer 4.0
http://msdn.microsoft.com/workshop/components/activex/buildax.asp

Using MSHTML
http://msdn.microsoft.com/workshop/browser/Hosting/Hosting.asp

Web Sites
http://www.microsoft.com/com/resources/websites.asp

Executing Files by Hyperlink and the File Download Dialog Box
http://support.microsoft.com/support/kb/articles/Q232/0/77.ASP

8. Users Who Administer Their Own Computers

If you do not have a system or network administrator to manage your computer, you can follow the suggestions in this section to improve the security of using ActiveX controls.

Use Windows Update and Office Update regularly.
It is important to use Microsoft's Windows Update to apply the most recent patches provided by Microsoft. The Windows Update mechanism (http://windowsupdate.microsoft.com) is an ActiveX control downloaded from Microsoft that checks for available updates to system files, device drivers, service packs, and new Windows features. In addition, the patches applied by Windows Update can disable, patch, or remove ActiveX controls that have been found to contain security risks. If you have Microsoft Office, you should use Microsoft's Office Update (http://officeupdate.microsoft.com) as well because Microsoft Office contains additional ActiveX controls that Windows Update does not maintain.

Since the Windows Update and Office Update product catalogs are tailored for the programs and components you have installed at the time you visit the Update page, you should check Windows Update and Office Update whenever you add additional components to your system. If you re-install Windows or Office, you also need to re-install any components you had previously downloaded through the Update mechanisms.

Configure and use security zones.
There are several choices for configuring security zones on your computer.

Restrict ActiveX controls in the Internet zone.
One way to prevent the exploitation of ActiveX controls is to limit their functionality in the Internet security zone (see the section for system administrators for more information about zones). If you require a greater level of trust and security, you may wish to apply some of these changes to other security zones as well. Below are recommendations for the settings of two important options:

1. *Run ActiveX controls and plug-ins.* You can disable the execution of ActiveX controls to protect against potentially malicious ActiveX controls, but disabling this option also prevents plug-ins from executing normally. Since plug-ins for common applications such as Adobe Acrobat are included in this same category,

you should be aware that setting the option to “disable” results in significantly reduced functionality. If you decide to set this option to “prompt” instead of disabling it, keep in mind that “prompt” not always an effective choice because it is not always clear what the safe response should be.

An excellent solution (but perhaps one requiring more administrative effort) is to set this option to “Administrator Approved.” In this setting, only the ActiveX controls approved by the administrator (using the Internet Explorer Administration Kit) will be executed. For more information regarding this option, see http://www.microsoft.com/Windows/ieak/en/support/faq/default.asp

2. *Script ActiveX controls marked safe for scripting.* You can disable the scripting of ActiveX controls marked “Safe for Scripting” to protect against many potential risks of ActiveX controls, but disabling this option limits the normal operation of many legitimate controls used over the Internet. Setting this option to “prompt” reduces the potential risk, but you must make a decision regarding the safety of the control.

Here are steps for changing your security zone settings for Internet Explorer 5.x:

1. Start Internet Explorer as you normally would.
2. From the **Tools** menu select **Internet Options....** The Internet Options dialog box appears.
3. Select the **Security** tab. The Security Options panel appears.
4. Select the zone you wish to change. For most users, this is the **Internet** zone, but depending on your circumstances, you may need to repeat these steps for the **Local Intranet** zone as well.
5. Click the **Custom Level** button. The Security Settings panel appears.
6. Change one or more of the following settings based on the information provided earlier and your desired level of security.
7. Set **Run ActiveX controls and plug-ins** to administrator approved, disable, or prompt.
8. Set **Script ActiveX controls marked safe for scripting** to disable or prompt.
9. Click **OK** to accept these changes. A dialog box appears asking if you are sure you want to make these changes.
10. Click **Yes**.
11. Click **Apply** to save your changes.
12. Click **OK** to close the Internet Options dialog box.

Use the Trusted Sites zone.
Security zones can also be used to enable ActiveX controls at specific sites where you wish to retain this functionality. To place a site in the Trusted Sites zone using Internet Explorer 5.x,

1. Start Internet Explorer as you normally would.
2. From the **Tools** menu select **Internet Options...**. The Internet Options dialog box appears.
3. Select the **Security** tab. The Security Options panel appears.
4. Select the **Trusted Sites** zone.
5. Click the **Sites...** button.
6. Enter the name of the trusted site in the **Add this Web Site to the zone:** text box.
7. Click the **Add** button.
8. If a dialog box appears saying “Sites added to this zone must use the https:// prefix. This prefix assures a secure connection":
   a. Click **OK**.
   b. Add https:// to the beginning of the site name, and try to add the site again.
   c. Or uncheck the box at the bottom of the dialog box marked **Require server verification (https:) for all sites in this zone.** Making this change reduces the security of your system by not requiring certificate-based authentication, relying instead on DNS-based verification, which could be misleading. We encourage you not to make this change unless you fully understand the implications. If you choose not to require certificate-based verification, you may wish to reduce other security settings for the Trusted Sites zone.
9. Click **OK** to save the new list of sites.
10. Click **Apply** to save your changes.
11. Click **OK** to close the Internet Options dialog box.

Exercise caution when prompted to run an ActiveX control.
In some cases, the security zone settings prompt the user before an ActiveX control is downloaded or run. Even with a patch installed, you may be given the option to allow a control to be executed. You should be aware of this—before you allow the control to run, consider the source that is supplying the control and the purpose of the control.

Disable ActiveX controls and other scripting languages in mail clients.
To operate your computer more securely, you should also change the settings in your email. Below is information about Microsoft Outlook, Outlook Express, and other mail clients.

Microsoft Outlook Security Update
The Outlook Security Update prevents malicious email from downloading or executing ActiveX controls. The Microsoft Outlook 2000 E-Mail Security Update modifies Outlook to use the Restricted zone and disables the use of Active Scripting in that zone. It also limits which attachment file types are displayed in Outlook messages and adds new prompts for accessing the address book or sending email messages. Unfortunately, this security patch may disable functionality in other programs that interact with Outlook. You need to consider the impact of this update before proceeding with installation.

More information about the Outlook 2000 E-Mail Security Update is available from http://www.officeupdate.com/2000/downloadDetails/Out2ksec.htm

Microsoft Outlook Express and Alternatives to the Outlook Security Update
For some, the patch provided by Microsoft is adequate. For others, particularly those using non-Microsoft networking products, the patch does not provide complete protection. Still others may find the changes imposed by the security patch too restrictive. If you do not wish to install the Outlook Security Patch, you can apply many of the changes manually.

Because an email message may start Internet Explorer automatically if Active Scripting is enabled, you should configure the Outlook email client to use the Restricted zone.

The steps for configuring Outlook to use the Restricted zone are

1. Start Outlook as you normally would.
2. From the **Tools** menu select **Options...**. The Options dialog box appears.
3. Select the **Security** tab. The Security Options panel appears.
4. In the **Secure content** section, change the pull-down menu from **Internet** to **Restricted Sites**.
5. Click **Apply** to save your changes.
6. Click **OK** to close the Options dialog box.

Because the Restricted zone still allows the execution of scripts, an intruder can send you an email message that, when viewed, starts Internet Explorer and immediately runs an ActiveX control. To protect against this scenario and others like it, you should disable Active Scripting in the Restricted zone.

The steps for disabling Active Scripting in the Restricted zone are

1. Start Internet Explorer as you normally would.
2. From the **Tools** menu select **Internet Options...**. The Internet Options dialog box appears.
3. Select the **Security** tab. The Security Options panel appears.
4. Select the **Restricted** zone.

5. Click the **Custom Level** button. The Security Settings panel appears.
6. Set **Active Scripting** to disable.
7. Click **OK** to accept these changes. A dialog box appears asking if you are sure you want to make these changes.
8. Click **Yes**.
9. Click **Apply** to save your changes.
10. Click **OK** to close the Internet Options dialog box.

Other Email Clients

If you use Internet Explorer as your web browser, you may wish to disable JavaScript and other scripting languages in your email client to prevent an email message from starting Internet Explorer and then running an ActiveX control.

A Last Word

Participants in the Security in ActiveX Workshop spent two intensive days on ActiveX controls and ways to use them in the safest way possible. This paper contains the outcome of that work. Though we have provided information for separate audiences, action by all groups—management, system administrators, security personnel, developers, and individual users—is needed to increase the safety of using ActiveX.

Workshop participants encourage readers to distribute this paper widely and also to be vigilant, keeping informed about further developments by checking the references listed in this report and monitoring web sites such as those of the CERT Coordination Center (www.cert.org) and Microsoft Corporation (www.microsoft.com/security/).

Appendix A – Reporting ActiveX Security Problems

The workshop participants encourage readers to use the following methods of reporting security problems so that the developer of the ActiveX control has an opportunity to fix the problem before details of the vulnerability are publicly distributed, increasing the chance of exploitation.

Reporting to the CERT/CC

If you identify a security problem related to an ActiveX control, please notify the CERT Coordination Center using the vulnerability reporting form that can be found at http://www.cert.org/reporting/vulnerability_form.txt. Include as much information as possible.

The CERT/CC sends the vulnerability information to vendors before making it public. (The CERT/CC vulnerability disclosure policy is available from http://www.cert.org/faq/vuldisclosurepolicy.html). If you wish to notify Microsoft or a third-party developer directly, please copy cert@cert.org on the message.

Reporting to Microsoft

To contact Microsoft, send a description of the security problem to secure@microsoft.com.

More information about this address can be found at www.microsoft.com/Technet/security/alertus.asp.
Please remember to send a copy of your message to cert@cert.org.

Appendix B – Known Vulnerabilities

The following information has been extracted from the notes that accompanied a presentation at the Security in ActiveX Workshop, "A Survey of ActiveX Security Problems." Though the speaker cautioned that it is not necessarily comprehensive nor, perhaps, 100 percent accurate, the notes served as a useful reference for discussing ActiveX vulnerabilities. The list of vulnerabilities was, as of the August 2000 workshop, believed to be reasonably complete in terms of identifying known vulnerabilities. Readers who are aware of a vulnerability in an ActiveX control that is not on this list should report it to the CERT Coordination Center (cert@cert.org).

The presenter provided as much as the following information as was possible at the time:

- the name of the control as it appears in OLEView
- the implementation file for the control
- the class ID
- the vendor
- pointers to relevant Microsoft security bulletins
- pointers to CERT advisories
- a description of the impact
- a statement about the safe-for-scripting status of the control
- a summary of the nature of the patch
- an indication of whether an exploit is publicly available

The controls listed below are grouped roughly by similar functionality to help distinguish between similar vulnerabilities. Information is marked with question marks if it was implied, not confirmed, or if it might be otherwise inaccurate.

Vulnerable ActiveX Controls

VU#9162	Wang/Kodak Image Edit Control imgedit.ocx {6D940280-9F11-11CE-83FD-02608C3EC08A} Vendor: Wang/Kodak/Microsoft? Microsoft Security Bulletin MS99-037, Sep 10 1999 Attacker can create local files Control was previously marked safe-for-scripting Patch sets kill bit A public exploit was not found, but is likely to exist
VU#23412	Wang/Kodak Image Annotation Control imgedit.ocx {6D940285-9F11-11CE-83FD-02608C3EC08A} Vendor: Wang/Kodak/Microsoft? Microsoft Security Bulletin MS99-037, Sep 10 1999 Attacker can create local files? Control was previously marked safe-for-scripting Patch sets kill bit A public exploit was not found, but is likely to exist
VU#41408	Wang/Kodak Image Scan Control imgscan.ocx {84926CA0-2941-101C-816F-0E6013114B7F} Vendor: Wang/Kodak/Microsoft?

Microsoft Security Bulletin MS99-037, Sep 10 1999
Attacker can create local files?
Control was previously marked safe-for-scripting
Patch sets kill bit
A public exploit was not found, but is likely to exist

VU#24839 Wang/Kodak Image Thumbnail Control
imgthumb.ocx {E1A6B8A0-3603-101C-AC6E-040224009C02}
Vendor: Wang/Kodak/Microsoft?
Microsoft Security Bulletin MS99-037, Sep 10 1999
Attacker can create local files?
Control was previously marked safe-for-scripting
Patch sets kill bit
A public exploit was not found, but is likely to exist

VU#26924 Wang/Kodak Image Admin Control
imgadmin.ocx {009541A0-3B81-101C-92F3-040224009C02}
Vendor: Wang/Kodak/Microsoft?
Microsoft Security Bulletin MS99-037, Sep 10 1999
Attacker can create local files?
Control was previously marked safe-for-scripting
Patch sets kill bit
A public exploit was not found, but is likely to exist

VU#38955 Object for constructing type libraries for scriptlets
scrobj.dll {06290BD5-48AA-11D2-8432-006008C3FBFC}
Vendor: Microsoft
Microsoft Security Bulletin MS99-032, Aug 31 1999
CVE Name: CVE-1999-0668
Attacker can create or modify local files
Widely exploited by KAK virus
Used in Bubbleboy exploit
Control was previously marked safe-for-scripting
Also can be referenced as Scriptlet.Typelib
Patch marks control as not safe-for-scripting
An exploit is publicly available (Guninski)

VU#15504 Scripting.FileSystemObject
scrrun.dll {0D43FE01-F093-11CF-8940-00A0C9054228}?
Vendor: Microsoft
Microsoft Security Bulletin not found
Attacker can read and write local files
Control safe-for-scripting status is unknown
Reported on May-5-1999
Patch status is unknown
An exploit is publicly available (Luzsicza)

VU#31356 Windows Media Player
msdxm.ocx {22D6F312-B0F6-11D0-94AB-0080C74C7E95}
Vendor: Microsoft
Microsoft Security Bulletin not found
Attacker can test for the existence of a local file
Control isn't marked as safe-for-scripting, but appears to be (unknown)
Patch is non-existent?
An exploit is publicly available (Guninski)
VU#31994 ActiveMovieControl Object

msdxm.ocx {05589FA1-C356-11CE-BF01-00AA0055595A}
Vendor: Microsoft
Microsoft Security Bulletin not found
CVE Name: CAN-2000-0400
Attacker can place arbitrary files on the local system
Posted publicly to BUGTRAQ on May 13 2000
Control isn’t marked as safe-for-scripting, but appears to be (unknown)
Patch is non-existent?
An exploit is publicly available (SecurityFocus)

VU#32686 ActiveMovieControl Object
msdxm.ocx {05589FA1-C356-11CE-BF01-00AA0055595A}
Vendor: Microsoft
Microsoft Security Bulletin not found
An attacker can cause the victim machine to lock up
Causes ActiveMovie Object to read from "file:///aux"
Control isn’t marked as safe-for-scripting, but appears to be (unknown)
Patch status is unknown
A public exploit was not found

VU#39965 DHTML Edit Control Safe for Scripting for IE5
dhtmled.ocx {2D360201-FFF5-11D1-8D03-00A0C959BC0A}
Vendor: Microsoft
Microsoft Security Bulletin MS99-011, Apr 21 1999
CVE Name: CVE-1999-0487
Attacker can read files with known names on local hard drive
Attacker can read information the user supplied to the control
Control is marked safe-for-scripting
Patch changes control to restrict actions based on domain
An exploit is publicly available (Guninski)

VU#24176 DHTML Edit Control Safe for Scripting for IE5
dhtmled.ocx {2D360201-FFF5-11D1-8D03-00A0C959BC0A}
Vendor: Microsoft
Microsoft Security Bulletin not found
Attacker can obtain information in other frames and clipboard
The previous fix for VU#39965 appears to be incomplete
Control is marked safe-for-scripting
Patch is non-existent?
Posted publicly on BUGTRAQ on Jul 14 2000
An exploit is publicly available (Guninski)

VU#29795 Hhopen Control
hhopen.ocx {130D7743-5F5A-11D1-B676-00A0C9697233}
Vendor: Microsoft
Microsoft Security Bulletin MS99-037, Sep 10 1999
Attacker may exploit buffer overflow and execute arbitrary commands
Control was previously marked safe-for-scripting
Patch sets kill bit
An exploit is publicly available (Hird)

VU#25249 HHCtrl Object
hhctrl.ocx {ADB880A6-D8FF-11CF-9377-00AA003B7A11}
Vendor: Microsoft
Microsoft Security Bulletin MS00-037, Jun 2 2000
CERT Advisory CA-2000-12, June 19 2000

Compiled help files can execute arbitrary shortcuts
Attackers can invoke help files from a web page
Control is marked safe-for-scripting
Patch requires web sites to open only local help files
The CERT/CC considers the current patch to be incomplete
An exploit is publicly available (Guninski)

VU#26057 Preloader Control?
iepreld.ocx {16E349E0-702C-11CF-A3A9-00A0C9034920}
Vendor: Microsoft
Microsoft Security Bulletin MS99-018, May 27 1999
CVE Name: CVE-1999-0917
Attacker can obtain a list of files on the local hard drive
Control safe-for-scripting status is unknown (no answer in bulletin)
Patch removes the control and removes the class id from the registry
Can this control be re-installed by an attacker? (KB Article: Q231452)
Also called the "Legacy ActiveX Control"

VU#35626 Microsoft Office UA Control
ouactrl.ocx {8936033C-4A50-11D1-98A4-00A0C90F27C6}
Vendor: Microsoft
Microsoft Security Bulletin MS00-034, May 12 2000
CERT Advisory CA-2000-07, May 24 2000
CVE Name: CVE-2000-0419
Attacker can script functions in MS Office 2000 products
Attacker can disable macro virus protection
Control is marked safe-for-scripting
Patch removes the ability to script office functions
Patch disables popup and showme functionality in help
An exploit is publicly available (?)

VU#8995 Microsoft Forms 2.0 TextBox
fm20.dll {8BD21D10-EC42-11CE-9E0D-00AA006002F3}
Vendor: Microsoft
Microsoft Security Bulletin MS99-001, Jan 21 1999
CVE Name: CVE-1999-0384
Attacker can read or export text on a user's clipboard
Control is marked safe-for-scripting
Patch corrects error without changing functionality
An exploit is publicly available (Cuartango)

VU#8995 Microsoft Forms 2.0 ComboBox
fm20.dll {8BD21D30-EC42-11CE-9E0D-00AA006002F3}
Vendor: Microsoft
Microsoft Security Bulletin MS99-001, Jan 21 1999
CVE Name: CVE-1999-0384
Attacker can read or export text on a user's clipboard
Control is marked safe-for-scripting
Patch corrects error without changing functionality
An exploit is publicly available (Cuartango)

VU#30423 Microsoft Scriptlet Component
mshtml.dll {AE24FDAE-03C6-11D1-8B76-0080C744F389}
Vendor: Microsoft
Microsoft Security Bulletin MS99-012, Apr 21 1999
Attacker can circumvent domain security policy in IE

Attacker can execute arbitrary javascript?
Control isn’t marked safe-for-scripting, but appears to be
An exploit is publicly available (Guninski)
Patch corrects vulnerability

VU#25919 Acrobat Control for ActiveX
pdf.ocx {CA8A9780-280D-11CF-A24D-444553540000}
Vendor: Adobe
Adobe Security Bulletin was not found
Vulnerable Version: < 1.3.188?
Attacker may exploit buffer overflow and execute arbitrary commands
This issue was first public on or around Aug 30 1999
Control is marked safe-for-scripting
Patch removes overflow?
An exploit is publicly available (Hird)
Can also be referenced as PDF.PdfCtrl.1

VU#1673 EYEDOG Control
eyedog.ocx {06A7EC63-4E21-11D0-A112-00A0C90543AA}
Vendor: Microsoft
Microsoft Security Bulletin MS99-032, Aug 31 1999
Attacker may exploit buffer overflow and execute arbitrary commands
Attacker can retrieve system configuration information
Control was previously marked safe-for-scripting
Patch sets kill bit
An exploit is publicly available (Hird)

VU#34453 "Launch" ActiveX Control?
?.? {????}
Vendor: SystemSoft
A SystemSoft security bulletin was not found
Attacker can execute arbitrary commands
Control was and may still be marked safe-for-scripting
Can also be referenced as LAUNCH.LaunchCtrl.1
Patch status is unknown
An exploit is publicly available (Smith)
Reported on or around: Jul 29 1999

VU#22919 "RegObj" ActiveX Control?
?.? {????}
Vendor: SystemSoft
A SystemSoft security bulletin was not found
Attacker can read and write registry keys
Control was and may still be marked safe-for-scripting
Can also be referenced as IISSample.RegistryAccess.1?
Patch status is unknown
A public exploit was not found (Richard Smith may know of one)
Reported on or around: Jul 29 1999

VU#3062 CEnroll Class
xenroll.dll {43F8F289-7A20-11D0-8F06-00C04FC295E1}?
Vendor: Microsoft
Microsoft Knowledgebase Article was found (Q242366)
Attacker can create arbitrarily named files
Control is marked safe-for-scripting
Patch appears to limit the number of times the control can be called

A public exploit was not found
Reported on or around: Jul 29 1999

VU#41233 BRAUTO Control
?.? {3C1A3E33-9566-11D0-A5C4-00608CC9A71F}
Vendor: Encompass Corp.
An Encompass security bulletin was not found (site was down?)
Attacker may be able to gain name, address, phone number, etc.
Control is marked safe-for-scripting until used
Can also be referenced as BRAUTO.BRautoCtrl.1
A public exploit was not found (Richard Smith may know of one)
Patch status is unknown
Reported on or around: Jul 29 1999

VU#22505 RegWizCtrl
regwizc.dll {50E5E3D1-C07E-11D0-B9FD-00A0249F6B00}
Vendor: Microsoft
Microsoft Security Bulletin MS99-037, Sep 10 1999
Attacker may exploit buffer overflow and execute arbitrary commands
Control was previously marked safe-for-scripting
Patch sets kill bit
An exploit is publicly available (Hird)

VU#40813 Setupctl 1.0 Type Library?
Setupctl.dll {F72A7B0E-0DD8-11D1-BD6E-00AA00B92AF1}
Vendor: Microsoft
Microsoft Security Bulletin MS99-037, Sep 10 1999
Attacker may exploit buffer overflow and execute arbitrary commands
Control is part of an old Internet Explorer Active Setup
Control safe-for-scripting status is unknown
Patch sets kill bit
An exploit is publicly available (Hird)

VU#31947 MSN ActiveX Setup BBS Control?
?.? 8F0F5093-0A70-11D0-BCA9-00C04FD85AA6
Vendor: Microsoft
Microsoft Security Bulletin was not found
Attacker may exploit buffer overflow and execute arbitrary commands
Control safe-for-scripting status is unknown
An exploit is publicly available (Hird)
Can also be referenced as MSN_SetupBBS.MSN Setup BBS

VU#39543 InstallEngineCtl Object
asctrls.ocx {6E449683-C509-11CF-AAFA-00AA00B6015C}
Vendor: Microsoft
Microsoft Security Bulletin MS99-048, Nov 11 1999
CVE Name: CVE-2000-0329
Attacker may interact with local cabinet files
Attacker may execute arbitrary commands
Control is marked safe-for-scripting
The control is sometimes called the Active Setup Control
Patch prevents control from accessing local unsigned cabinet files
An exploit is publicly available (SecurityFocus)

VU#35633 InstallEngineCtl Object

	asctrls.ocx {6E449683-C509-11CF-AAFA-00AA00B6015C} Vendor: Microsoft Microsoft Security Bulletin MS00-042, Jun 29 2000 CVE Name: CAN-2000-0160 Issue posted publicly to BUGTRAQ on Feb 19 2000 (Cuartango) Always trusts MS signed cab files and installs files Attacker can install Microsoft signed components on the victim’s system? Control is marked safe-for-scripting The control is sometimes called the Active Setup Control Patch causes Microsoft content to be treated like other vendors A public exploit was not found
VU#22652	InstallEngineCtl Object asctrls.ocx {6E449683-C509-11CF-AAFA-00AA00B6015C} Vendor: Microsoft Microsoft Security Bulletin MS00-042, Jun 29 2000 Attacker can overwrite local files (file permissions permitting) Control is marked safe-for-scripting Patch prevents caller from specifying the file location A public exploit was not found
VU#28370	Taskpads ActiveX Control? ?.? {D306C3B7-2AD5-11D1-9E9A-00805F200005}? Vendor: Microsoft Microsoft Security Bulletin MS99-007, Feb 22 1999 Control was previously marked safe-for-scripting Attacker may run local executables? Patch removes taskpad functionality (sets the kill bit?) A public exploit was not found
VU#23887	Shell DefView shell32.dll {1820FED0-473E-11D0-A96C-00C04FD705A2} Vendor: Microsoft Microsoft Security Bulletin not found Local attacker can execute programs when a victim browses a folder as a web page Default action on InvokeVerb method is Open (Execute) Control was not marked safe-for-scripting? Patch is non-existent? An exploit is publicly available (Guninski)
VU#37466	Microsoft NetShow Player msdxm.ocx {2179C5D3-EBFF-11CF-B6FD-00AA00B4E220} Vendor: Microsoft Microsoft Security Bulletin not found Control is not marked safe-for-scripting, but appears to be Attacker may crash victims system Patch status unknown No public exploit was found
VU#22979	Outlook Express Message List msoe.dll {233A9692-667E-11D1-9DFB-006097D50408} Vendor: Microsoft Attacker can obtain the number of messages in a folder Control is marked safe-for-scripting Patch status unknown No public exploit was found

Active X Vulnerabilities (Incomplete Data)

There appears to be a vulnerability in each of these controls. The descriptions of the vulnerabilities listed in this section were incomplete at the time the workshop notes were printed.

VU#40210	hpsocsc ActiveX Control (from HP)
VU#40193	RU3.Ru3Ctrl.1
VU#22507	Office Documents act as ActiveX controls? Microsoft Security Bulletin MS00-049, Jun 13 2000 Powerpoint and Excel can execute VBA code Posted publicly to BUGTRAQ on Jun 17 2000 An exploit is publicly available (Guninski) Controls were previously marked as safe-for-scripting Patch marks the controls as not safe-for-scripting
VU#34485	Microsoft Word Posted publicly to BUGTRAQ on Aug 7 2000 Word can open Access databases for Mail Merge Access can then execute arbitrary VB code An exploit is publicly available (Guninski)
VU#27857	IE can open Access databases and execute VB code Microsoft Security Bulletin MS00-049, Jul 13 2000 CVE Name: CAN-2000-0596 CERT Advisory CA-2000-16, Aug 11 2000 Posted publicly to BUGTRAQ on Jun 17 2000 An exploit is publicly available (Guninski) This vulnerability is also called the “IE Script” vulnerability
VU#41376	Microsoft Scriptlet Rendering ActiveX Control (from Microsoft) Microsoft Security Bulletin MS00-055, Aug 09 2000 Attacker can read, but not add, change or delete local files control will "render" files of any type, not just HTML
VU#32531	WUSysInfo.WUSysInfo.1 = Windows 98 update control Windows 98 Update control

Other ActiveX-Related Vulnerabilities

VU#27910	Microsoft signed ActiveX Controls can be installed without prompts Reported: Feb-19-2000 Microsoft Security Bulletin MS00-042, Jun 19 2000
VU#5583	ActiveX controls have access to user's private key
VU#34414	Buffer overflow in IE code to instantiate ActiveX controls Microsoft Security Bulletin MS00-033, May 17 2000 CVE Name: CVE-2000-0464 Also called the “Malformed Component Attribute” vulnerability
VU#16597	RDS MDAC

	Microsoft Security Bulletin MS98-004, Jul 17 1998
	Microsoft Security Bulletin MS99-025, Jul 19 1999
VU#38950	Files created outside cache directory
	Microsoft Security Bulletin MS0-0046, Jul 20 2000

Notes on ActiveX Control Counts

In preparing for the Security in ActiveX Workshop, the speaker surveyed several machines. Below are his findings.

On my Win2K professional system: IE 5.5, Office 2000

2565 objects
144 controls
52 controls safely initializable from persistent data
46 controls marked safe-for-scripting

On my NT4 workstation system: IE 5.0, Office 97

1295 objects
94 controls
54 controls safely initializable from persistent data
64 controls marked safe-for-scripting

On my wife's Windows 98 system: IE 5.0, Office 97

1141 objects
84 controls
48 controls safely initializable from persistent data
66 controls marked safe-for-scripting

Other Interesting Controls

Windows Script Host Shell Object
wshom.ocx {F935DC22-1CF0-11D0-ADB9-00C04FD58A0B}
Used in Scriptlet.Typelib exploit to run command.com
Also can be referenced as Wscript.Shell

FileSystem Object
scrrun.dll {0D43FE01-F093-11CF-8940-00A0C9054228}
Used in exploits to create local files
Also can be referenced as Scripting.FileSystemObject

Microsoft Excel Worksheet
excel.exe {00020820-0000-0000-C000-000000000046}

Microsoft Excel Chart
execl.exe {00020821-0000-0000-C000-000000000046}

Microsoft PowerPoint Presentation
powerpnt.exe {64818D10-4F9B-11CF-86EA-00AA00B929E8}

Microsoft PowerPoint Slide

powerpnt.exe {64818D11-4F9B-11CF-86EA-00AA00B929E8}

Microsoft Word Document
winword.exe {00020906-0000-0000-C000-000000000046}

Microsoft Access Application
msaccess.exe {73A4C9C1-D68D-11D0-98BF-00A0C90DC8D9}

Controls safely initializable from persistent data
{7DD95802-9882-11CF-9FA9-00AA006C42C4}
409 = Controls safely initializable from persistent data
800 = Safe for initialization

Controls that are safely scriptable
{7DD95801-9882-11CF-9FA9-00AA006C42C4}
409 = Controls that are safely scriptable
800 = Safe for scripting

References

http://www.nat.bg/~joro
http://home.ntware.com/bugs/activex_bug__1.html
http://home.ntware.com/bugs/activex_bug__2.html
http://home.ntware.com/bugs/activex_bug__3.html
http://home.ntware.com/bugs/activex_bug__4.html
http://home.ntware.com/bugs/activex_bug__5.html
http://home.ntware.com/bugs/activex_bug__6.html
http://www.tiac.net/users/smiths/acctroj/hp.htm
http://www.securityfocus.com/data/vulnerabilities/exploits/775.html

Appendix C – Other Sources of Information

Internet Explorer Administration Kit (IEAK) information
http://www.microsoft.com/Windows/ieak/en/support/faq/default.asp

Internet Explorer security zones (configuring zones and creating new zones)
http://msdn.microsoft.com/workshop/security/szone/urlzones.asp

IObjectSafety information, along with sample code
http://msdn.microsoft.com/workshop/components/com/IObjectSafetyExtensions.asp

Mobile code information
http://www.microsoft.com/technet/security/mblcode.asp

Fred McLain's original write-up on his "Exploder" ActiveX control
http://www.halcyon.com/mclain/ActiveX/Exploder/FAQ.htm

A write-up on dangerous ActiveX controls that ship with Windows 98 PCs
http://users.rcn.com/rms2000/acctroj/

Checking for dangerous ActiveX controls being installed on a system:
http://users.rcn.com/rms2000/acctroj/axcheck.htm

See also the references in the report sections for particular groups of readers and in Appendix B.

This document was last updated on January 3, 2001 (links updated)