ITL Bulletin

ADVISING USERS ON INFORMATION TECHNOLOGY

SECURITY IMPLICATIONS OF ACTIVE CONTENT

By Wayne Jansen and Tom Karygiannis
Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology

Introduction

In today's world, both private and public sectors depend upon information technology (IT) systems to perform essential and mission-critical functions. Often, as technology improves to provide new capabilities and features, new vulnerabilities are introduced along with these functional improvements. Organizations implementing and using these advanced technologies must, therefore, be increasingly on guard.

One such emerging technology is *active content*. Although the term has different connotations among individuals, it is used here in its broadest sense to refer to electronic documents that, unlike ASCII character documents of the past, are able to automatically carry out or trigger actions without the intervention of a user. Examples of active content include PostScript® documents, Java™ applets, JavaScript™, word processing and spreadsheet macros, and executable electronic mail attachments.

The purpose of this bulletin is to provide an overview of this technology so that the reader is better informed about the associated security risks and can make more informed IT security decisions. The bulletin provides real-world examples involving commonly available products and development tools as a way of increasing the understanding and awareness of the potential risks involved. A glossary of relevant terms and links to useful online references are also included at the end of this publication.

Background

Having the ability to download files and electronic documents off the Internet is a useful function and a common practice for many people today. While there are risks involved if one visits an unknown site, it appears at first glance that there should be no harm in downloading information as long as the files are non-executables. Even if a browser plug-in or utility is downloaded, it is recognized as such and must be explicitly installed in order to function, so careful judgment and appropriate preparation can be taken in advance. This view on risks, however, is incorrect**.** Today, electronic documents are themselves programs or contain programs that can be self-triggered. Loading a document into a word processor can produce the same effect as executing a program, requiring appropriate caution to be taken. After all, if you would not knowingly execute a program from an unknown source, why would you indirectly execute one embedded in an electronic document?

In striving to offer greater functionality and flexibility, software developers will continue to blur the distinctions between program and data. While the developer's intentions are presumably good, they can often have a negative impact when the need for security is not fully taken into account. Such documents are said to have active content, which involves new technology such as built-in macros, scripting languages, and virtual machines. The trend towards active content has been spurred by the popularity of the Web. A dynamic weather map, a stock ticker, and live camera views or programmed broadcasts appearing on a Web page are common examples of how this technology is being applied. Like any technology, active content can provide a useful capability, but can also become a source of vulnerability for an attacker to exploit.

Continued on page 2

ITL Bulletins are published by the Information Technology Laboratory (ITL) of the National Institute of Standards and Technology (NIST). Each bulletin presents an in-depth discussion of a single topic of significant interest to the information systems community. **Bulletins are issued on an as-needed basis** and are available from ITL Publications, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900, telephone (301) 975-2832. To be placed on a mailing list to receive future bulletins, send your name, organization, and address to this office.

Bulletins issued since November 1998

Active content can be considered to be a form of mobile code, whereby the code in the form of a script, macro, or other portable representation can move either indirectly (e.g., via an electronic mail attachment) or directly (e.g., via a Web page) from one platform to another where it eventually executes. Because the technology is designed to be seamless, a user is often unaware of what is happening. The problem with mobile code in general, and active content in particular, is that it can embed a Trojan horse or other form of malicious code into a system.

The prevalence of unintentional implementation errors in software applications that process electronic documents plagues active content technology. Even if a design is correct and secure, the implementation may unintentionally contain a serious vulnerability that can be exploited by malicious code conveyed in the active content portion of a document. Together, active content and implementation errors can damage or subvert an IT system. An attacker needs only to learn what software their target is using, find an appropriate exploit, and send the document to the target.

The trend in application software development is to add more features and greater complexity to products. Complexity begets more code, more code interacts with other code and, hence, more implementation errors occur. This trend, combined with the competitive pressures facing manufacturers to be first to the market, the technical and cost barriers to extensive testing, and a marketplace that chooses functionality over security, ensures attackers continual opportunities in the future.

Technology-Related Risks

Some popular active-content technologies are described below. They are provided as present-day examples and do not imply an endorsement of the technology or product by NIST. A number of them share the characteristic of providing a useful capability when used in a Web browser environment. However, an attacker can exploit them. The motivation for these technologies is to improve functionality and gain flexibility for the user. In a Web application, this involves moving code processing away from the Web server onto the client's Web browser. Allowing remote systems to run arbitrary code on a local system, however, poses serious security risks. Traditional client-server systems do not involve such risks since they rely on static code on both the server and client sides.

PostScript®

One of the earliest examples of active content is PostScript® document representation, still in wide use today. PostScript® is a page description language that is widely used on most computer platforms. It is the de facto standard in commercial typesetting and printing houses. PostScript® commands are language statements in ASCII text that are translated into the printer's machine language by a PostScript® interpreter built into the printer. The interpreter uses scalable fonts, eliminating the need to store a variety of font sizes. A PostScript® file contains a document description, which is specified in the PostScript® page description language. The language is a powerful interpreted language, comparable to many programming languages, and, therefore, PostScript® documents inherently convey active content. For example, the language defines primitives for file manipulation, which can be used in a PostScript® document to modify arbitrary files when the document is displayed or printed.

An early exploit of PostScript® technology involved the language's ability to set a password held by the interpreter. In some hardware implementations of the language interpreter, if the password was set, it remained in non-volatile memory and prevented subsequent documents from being printed unless they contained the same password. An attacker sending a password-setting document could disable the printer in this way, requiring hardware replacement to rectify the situation. Some PostScript® interpreters can be set to disable potentially harmful primitives, but such actions can also inhibit useful functions. This dilemma is a recurring theme with active content.

Who we are

The Information Technology Laboratory (ITL) is a major research component of the National Institute of Standards and Technology (NIST) of the Technology Administration, U.S. Department of Commerce. We develop tests and measurement methods, reference data, proof-of-concept implementations, and technical analyses that help to advance the development and use of new information technology. We seek to overcome barriers to the efficient use of information technology, and to make systems more interoperable, easily usable, scalable, and secure than they are today.

Java™

Java is a full-featured programming language compiled into platform-independent byte code that is executed by an interpreter called the Java™ Virtual Machine. The resulting byte code can be executed where compiled or transferred to another Java-enabled platform (e.g., conveyed via an HTML Web page as an applet). Java is useful for adding functionality to Web sites and many services offered by various popular Web sites require the user to have a Java-enabled browser. The developers of Java tried to address the problem of security and were successful to a large extent. Java code is confined to a *sandbox* that restricts the access of the code to computational resources based on its permissions. Permissions are assigned primarily based on the source of the code (where it came from) and the author of the code (who developed it).

The Java sandbox is designed to prevent Web applets from inspecting or changing files on a client file system and using network connections to circumvent file protections or people's expectations of privacy. Hostile applets, however, still pose security threats even while executing within the sandbox. A hostile applet can consume or exploit system resources in an inappropriate manner or cause a user to perform an undesired or unwanted action. Examples of hostile applets exploits include denial-of-service, mail forging, invasion of privacy (e.g., exporting of identity, electronic mail address, and platform information) and installing backdoors to the system. The Java security model is fairly complex and can be difficult for a user to understand and manage, which can

increase risk. Moreover, many implementation bugs have also been found, which allow one to bypass security mechanisms.

JavaScript™ and Visual Basic® Script

JavaScript is a general-purpose, cross-platform scripting language whose code can be embedded within standard Web pages to create interactive documents. JavaScript, which is similar to Microsoft® Jscript®, was developed by Netscape. Both are founded on the same standard, the ECMAScript Language Specification, ECMA-262. The scripting language is extremely powerful and able to perform anything a user can do within the context of the browser. Design and implementation bugs have been discovered in both commercial scripting products. JavaScript does not have methods for directly accessing a client file system or for directly opening connections to other computers besides the host that provided the content source.

Visual Basic® Script (VBScript) is a programming language developed by Microsoft® for creating scripts that can be embedded in Web pages for viewing with the Internet Explorer browser. VBScript is a subset of the widely used Microsoft® Visual Basic® programming language and also works with Microsoft® ActiveX® Controls. The language is similar to JavaScript and poses similar risks.

In theory, confining a scripting language to boundaries of a Web browser should provide a relatively secure environment. In practice, this has not been the case. The main sources of problems have been twofold: the prevalence of implementation flaws and the close binding of the browser to related functionality such as an electronic mail facility or the underlying operating system. Past exploits include sending a user's URL history list to a remote site, and stealing the mail address of the user and forging electronic mail.

ActiveX®

ActiveX® is a set of technologies from Microsoft® that provide tools for linking desktop applications to the Web. ActiveX® controls are reusable component program objects that can be attached to electronic mail or downloaded from a Web site. ActiveX® controls also come preinstalled on Windows® platforms. Unlike Java, which is a platform-independent programming language, ActiveX® controls are distributed as executable binaries and must be separately compiled for each target machine and operating system.

The ActiveX® security model is considerably different from the Java sandbox model. The Java model restricts the permissions of applets to a set of safe actions. ActiveX®, on the other hand, places no restrictions on what a control can do. Instead, ActiveX® controls are digitally signed by their author under a technology scheme called Authenticode. The digital signatures are verified using identity certificates issued by a trusted certificate authority to an ActiveX® software publisher. For an ActiveX® publisher's certificate to be granted, the software publisher must pledge that no harmful code will be knowingly distributed under this scheme. The Authenticode process ensures that ActiveX® controls cannot be distributed anonymously and that tampering with the controls can be detected. This certification process, however, does not ensure that a control will be well behaved. The ActiveX® security model assigns the responsibility for the computer system's security to the user.

Before the browser downloads an unsigned ActiveX® control or a control whose corresponding publisher's certificate was issued by an unknown certifying authority, the browser presents a dialog box warning the user that this action may not be safe. Users can choose to abort the transfer or may continue the transfer if they assume the source is trustworthy or they are willing to assume the risk. Users may not be aware of the security implications of this decision, which may result in poor decisions. Even when the user is well informed, attackers may trick the user into approving the transfer. In the past, attackers have exploited implementation flaws to cover the user dialogue window with another that displays an unobtrusive message such as "Do you want to continue?" while exposing the positive indication button needed to launch active content. Recent versions of Internet Explorer allow the user to customize the behavior of ActiveX® controls depending on whether they are downloaded from a site on the Internet, a site on the local area network, or a site belonging to sets of identified trusted and untrusted sites.

Desktop Application Macros

Developers of popular spreadsheet, word processing, and other desktop applications created macros to allow users to automate and customize repetitive tasks. A macro is a series of menu selections, keystrokes, and commands that have been recorded and assigned a name or key combination. When the macro name is called or the macro key combination is pressed, the steps in the macro are executed from beginning to end. Macros are used to shorten long menu sequences as well as to create miniature programs within an application. Macro languages often include programming controls (IF, THEN, GOTO, WHILE, etc.) that automate sequences like any programming language. A virus can be written into a macro that is stored in a spreadsheet or word processing document. When the document is opened, the macro is executed and the virus is activated. It can also attach itself to subsequent documents that are saved with the same macro.

The recent Melissa virus is an example of the potential risk involved. A Microsoft® Word document containing a malicious Visual Basic® for Applications macro propagated itself through the Internet by sending the host document as an electronic mail attachment addressed to contacts found in the victim's address book. The newer generation of electronic mail applications, including the ones built into Web browsers, allows mail attachments to contain active content such as document macros or JavaScript programs. Since active content provides a number of avenues for exploits, such enclosures should be opened only after due consideration of the inherent risks.

Plug-Ins

Plug-ins are programs that work in conjunction with software applications to enhance their capabilities. Plug-ins are often added to Web browsers to enable them to support new types of content (audio, video, etc.). Such plug-ins can be downloaded from either the browser vendor's site or from a third-party site. Browsers typically prompt the user to download a new plug-in when a document that

requires functionality beyond the browser's current capabilities. Although plug-ins allow browsers to support new types of content, they are not active content in and of themselves, but simply an active-content-enabling technology. Windows® Media Player, RealPlayer, ThingViewer, QuickTime, Shockwave, and Flash are all examples of plug-ins that allow browsers to support new content types ranging from audio, video, interactive animation, and other forms of "new media."

From a security standpoint, plug-ins contain executable code and, therefore, precautions should be exercised in obtaining and installing them, as with any other software application. Downloading free software code and authorizing its installation by simply clicking an "Install now" or equivalent button is risky. Downloading plug-ins from reputable manufacturers can mitigate the risk, but even in this case, it is difficult for the user to always be aware of the security implications. In the past, unwanted side effects such as changes to browser security settings and tracking of a user's content preferences, however well intentioned, have occurred. Plug-ins designed to animate cursors or hyperlinks have also been designed to better track user preferences and viewing habits across a particular Web site. Although these additional capabilities may improve the user's experience with a particular Web site, the privacy and security implications are often not readily disclosed. Even if the site has a valid identity certificate associated with the signed downloaded code, this only tells the user that the manufacturer of the code has been verified by a certificate authority, but not whether the code obtained from them will behave non-maliciously or correctly. Users of plug-ins should be cautioned to read the fine print before agreeing to download executables and to take adequate measures to back up their system in case there are problems.

ITL Bulletins Via E-Mail

We now offer the option of delivering your ITL Bulletins in ASCII format directly to your e-mail address. To subscribe to this service, send an e-mail message to listproc@nist.gov with the message **subscribe itl-bulletin**, and your proper name, e.g., John Doe. For instructions on using listproc, send a message to listproc@nist.gov with the message **HELP**. To have the bulletin sent to an e-mail address other than the From address, contact the ITL editor at 301-975-2832 or elizabeth.lennon@nist.gov.

Countermeasures

A number of steps can be taken to mitigate the risks in using active content. The following sections highlight some of the more useful countermeasures one can apply.

Security Policy

Having or establishing an organizational security policy is an important first step in applying countermeasures for active content. For example, an Internet security policy can address enabling Java, JavaScript, or ActiveX® on individual user's Web browsers in various ways:

- Functionality must be disallowed completely;
- Functionality is allowed, but only from internal organizational servers;
- Functionality is allowed, but only from trusted external servers; and
- Functionality is allowed from any server.

If the policy is not stated clearly and consistently, and not made known throughout an organization, it creates a situation ripe for exploitation.

Application Settings

The desktop applications that handle documents containing active content typically have built-in controls that can be used to control or prevent access. For example, both Netscape and Microsoft® Web browsers have an options menu that can be used to select appropriate security settings regarding active content within downloadable documents. Spreadsheet, word processor, and presentation graphic software applications have similar controls. Even today, many manufacturers deliver products with insecure default settings. Users of such applications need to become familiar with the security options available and use them in accordance with organizational policy.

Automated Filters

If malicious content has been identified and understood, it can be detected and eliminated or rejected completely from entering. For example, many firewalls are capable of filtering electronic mail attachments for well-known file types, such as .exe executable files, and deleting them at the point of entry. More sophisticated filters can perform checks for viruses within executables and hidden macros within document files. Anti-virus software has also become increasingly capable of detecting electronic documents having active content with malicious code.

Version Control

Users can gain better security by applying security patches when available. This is a well-known and effective remedy, but for a variety of reasons, it is also an often-ignored one. Users can also take advantage of security enhancements to their applications by upgrading to newer versions. Microsoft® Windows® 98 users can use the Windows® update feature to find bug fixes and product updates and download them automatically from the Web. Using this feature, however, requires the user to use code that will scan the computer for installation information in order to properly install any upgrades. The Microsoft® Web site states that none of this information is sent to Microsoft® or over the Internet. Updating software products automatically over the Web is becoming increasingly popular, as the benefits to the user are considerable. As this practice becomes more commonplace, users must be better aware of the implicit decisions they make when allowing vendors to run software on their machine. For example, updating an audio player on a computer may allow the vendor to track the user's musical preferences.

Readers

Occasionally, manufacturers of desktop applications provide free software readers, which are capable of interpreting their proprietary file formats, for recipients of the documents who do not own the application. The Adobe® Acrobat® Reader, for example, allows users to view and print Portable Document Format (PDF) files, but does not allow users to edit them. Since the software readers are only

intended to produce a readable rendition of the document and are not full-fledge applications, they bypass many potentially harmful features and exploits based on implementation vulnerabilities contained in a specific application. Besides manufacturer-provided readers, general-purpose software readers are commercially available, which are capable of rendering dozens of file formats. A related measure is the selection of documents with less capable formats of active content, when multiple choices are offered. Some Web sites offer an electronic document in a variety of formats such as native word processor format, PostScript®, or PDF. While PDF represents text and graphics using the imaging model of the PostScript® language, PDF is not a programming language and contains no language constructs, making it the safest alternative.

System Isolation

Isolation works two ways. First, a production computer system that is unable to receive documents containing active content is unlikely to be affected by malicious hidden code. Although it is not always possible to isolate a system physically, logical isolation may be applied, at least to some degree. Second, risky functions, such as Web browsing, may be confined to a second system designated exclusively for that purpose. Often older or spare systems are available and could be put to good use this way.

Summary

Active-content documents offer several benefits to both the users of these documents and their authors. Java applets, JavaScript, and ActiveX® provide additional functionality to Web pages, plug-ins enable browsers to support new types of content, PostScript® offloads the processing and interpretation of the presentation of documents to the printer, and macros automate repetitive word processing and spreadsheet tasks. The benefits of each of these active-content technologies must be carefully weighed against the risks they pose. Security is not black or white, but shades of gray. When employing active-content technology, security measures should be put in place to reduce risk to an acceptable level and to recover if an incident occurs.

Informed security officers, administrators, and other IT professionals are responsible for developing security policies based on their organization's specific security needs and level of acceptable risk. Unfortunately, there is rarely a "one size fits all" guideline that fits the unique needs of every organization and each organization must decide what constitutes an acceptable level of risk. Establishing an organizational security policy is an important step in developing and applying appropriate security measures. The IT and security staff have a responsibility for keeping abreast of the associated risks with emerging technologies by subscribing to security mailing lists and visiting vendor Web sites for information and updates for products used within their organization. As active content moves beyond desktop personal computers to mobile handsets, television sets, and a wide variety of other consumer electronic goods, users will be faced with competing and difficult tradeoffs between privacy and security, with increased functionality and ease-of-use.

Before handling documents containing active content, consider seriously the following checklist, which summarizes some recommendations drawn from the previous material:

- Identify critical information resources and maintain regular backups.
- Identify and assess the risk to critical information resources from active content.
- Develop (or follow) the enterprise security policy regarding active content.
- Evaluate and install virus scanners, firewalls, and active-content filters according to enterprise security requirements. Keep these products upgraded to the latest version.
- Become knowledgeable of the security settings of desktop applications.
- Keep informed of the latest security advisories from CERT and subscribe to security mailing lists.
- Obtain and install the latest software upgrades and patches that address security vulnerabilities in desktop applications, such as Web browsers, readers, and electronic mail.
- Obtain all software through approved distribution channels.
- Institutionalize the download, evaluation, and distribution of needed plug-ins and freeware from the Internet to the organization.
- Read the fine print before agreeing to download application software and plug-ins.
- Do not run active content or software from untrusted sources. Enable ActiveX® code only from trusted Web sites that require its use.
- Consider using an isolated system and safe browser settings when visiting untrusted Web sites.
- Do not open documents containing active content or execute any electronic mail attachments without first verifying them with the sender. Be especially wary of attachments to electronic chain mails that have been forwarded from friends of friends.

Glossary

The following definitions highlight key concepts used throughout this bulletin:

- *Active Content*: Active content refers to electronic documents that are able to automatically carry out or trigger actions without the intervention of a user.
- *Computer Virus*: A computer virus is similar to a Trojan horse (see below) insofar as it is a program that contains hidden code, which usually performs some unwanted function as a side effect. The main difference is that the hidden code in a computer virus can replicate by attaching a copy of itself to other programs and may also include an additional "payload" that triggers when specific conditions are met.
- *Interpreter*: An interpreter is a program that processes a script or other program expression and carries out the requested action, in accordance with the language definition.
- *Malicious Code*: Malicious code refers to programs that are written intentionally to carry out annoying or harmful actions. They often masquerade as useful programs or are embedded into useful programs, so that users are induced into activating them. Types of malicious code include Trojan horses and computer viruses.

■ *Script*: A script is a sequence of commands, often residing in an ASCII file that can be interpreted and executed automatically. Unlike compiled programs, which execute directly on a computer processor, a script must be processed by another program that carries out the indicated action.

■ *Scripting Language*: A scripting language defines the syntax and semantics for writing scripts. Typically, scripting languages follow the conventions of a simple programming language, but they can also take on a more basic form such as a macro or a batch file. JavaScript, VBScript, and Perl are examples of scripting languages.

■ *Trojan Horse*: A Trojan horse is a useful or seemingly useful program that contains hidden code of a malicious nature. When the program is invoked, so is the undesired function whose effects may not become immediately obvious. The name stems from an ancient exploit of invaders' gaining entry to the city of Troy by concealing themselves in the body of a hollow wooden horse, presumed to be left behind by the invaders as a gift to the city.

Online Resources

A wealth of security information is available online. The following are a few notable sites that one can begin to explore for further information:

■ Microsoft® Internet Explorer Security Page. Microsoft® posts information and code fixes for security problems here as soon as they are available.
http://www.microsoft.com/windows/ie/security/default.asp

■ Netscape Security Page. Latest news concerning the security of Netscape's client, server, and development software.
http://home.netscape.com/security/notes/

■ Computer Security Resource Clearinghouse (CSRC). The CRSC contains current U.S. security policy documents, calendar of events, security publications, training resources, and information on various computer security subjects.
http://csrc.nist.gov

■ The Federal Computer Incident Response Capability (FedCIRC). FedCIRC provides a government focal point for incident reporting, handling, prevention and recognition.
http://www.fedcirc.gov/

■ WWW Security FAQ. The World Wide Web Consortium site contains a repository of information about the World Wide Web for developers and users.
http://www.w3.org/Security/Faq/

■ System Administration, Networking, and Security (SANS) Institute. The SANS community creates four types of products and services: system and security alerts and news updates, special research projects and publications, in-depth education, and certification.
http://www.sans.org

■ Computer Emergency Response Team (CERT) Coordination Center. CERT issues security advisories, helps start other incident response teams, coordinates the efforts of teams when responding to large-scale incidents, provides training to incident response professionals, and researches the causes of security vulnerabilities.
http://www.cert.org/

■ RISKS forum. ACM Committee on Computers and Public Policy forum on risks to the public in computers and related systems.
http://catless.ncl.ac.uk/Risks/

Disclaimer: Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology nor does it imply that the products mentioned are necessarily the best available for the purpose.

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
100 Bureau Drive, Stop 8900
Gaithersburg, MD 20899-8900

Official Business
Penalty for Private Use $300

Address Service Requested

PRSRT STD
POSTAGE & FEES PAID
NIST
PERMIT NUMBER G195