Global Information Assurance Certification Paper

Active Content - Administrators Beware

Russel Sanders
May 2001

Introduction:

The purpose of this paper is to construct a basic understanding of the threat that active content poses to information security. It is beyond the scope of this paper to measure the risk within your organization to these threats. Vulnerability is dependent upon several interrelated factors when it comes to active content. It is not just the vulnerability of the underlying software that allows active content to be successful; usually the attacker must rely upon a larger more uncontrolled variable. The larger variable I refer to is the human factor, which can be a great asset to the organization but can also be an unwitting ally of the attacker.

Several recommendations will be given in the form of rules that may help to mitigate the risk within your organization. No one thing can prevent the success of an active content assault and it is important to remember that these exploits pose threats to all of the fundamental aspects of information security which includes confidentiality, integrity and availability. So without further ado lets begin.

A cynical lawyer friend of mine once told me, "No good deed shall go unpunished." Never has this been truer than in the world of information technology. Take for instance active content; the idea was to create software that offers greater functionality and flexibility. This was to be accomplished by creating software that could functionally change how it works "automagically" with little or no interaction from the user of the software. For instance based upon a file that an end user may be trying to preview the browser might determine a plugin is required. These plugins represent an active content enabling technology that can instantly transform the browser into a viewer for sophisticated three-dimensional CAD drawings, or other content as required. While the intentions were well meaning the implementations have met with many problems made much worse by the fact that most of the active content tools were created without the forethought for security.

As early as 1997 authors like Tad Lane, and John Stewart were warning that active content posed a threat to information security, and it wasn't long afterward, their warnings were proven to be correct by exploiters such as Melissa and ExploreZip. Since these early articles we have suffered from many attacks that have been both benign and malignant affecting every aspect of information security.

Earlier than even these predictions, active content was causing problems for network administrators. One of the earliest scripting languages, Postscript® provided the attacker with an opportunity to create mayhem in the enterprise by sending codes within this page description language that would alter passwords on the receiving device. Normally these devices are printers, and they would obligingly write the password to NVRAM, unfortunately any subsequent print jobs that didn't use the password couldn't print. The only remedy was to swap out the affected component, which meant the printers would be unavailable until the switch was made. Some systems can still be afflicted by this exploit and for that it is important that you know and understand the hardware and software that have been implemented on your networks in order to better protect your corporate assets.

Overview of Modern Problems

Who hasn't succumbed to the lure of the most pervasive network ever created? Myself, I've become an information junkie constantly looking for dealers among the 1's and 0's of a binary nirvana. This however has not come without a great deal of risk, which can only be mitigated if you understand the problem.

The issue is not one of uniformed administrators it is usually a problem faced by ill informed and untrained end users. It is applications and operating systems installed with unsafe defaults. It is the blind indifference with which end users treat dialogue boxes that appears to inform them of impending doom, when they are merely trying to produce some meaningful work. And it is not providing filtered connections that strip harmful or unsigned content before it arrives at its unsuspecting targets.

To better understand the risks one must first understand that the proposed benefits are meant to outweigh the risks. Only your internal policies and organizational needs should determine if this is the case. In a wonderful world everyone would understand that when a dialogue box appears that you are meant to read the message and make an informed decision. If there is no policy that addresses the issue of interpretation of dialogue boxes that refers the reader to a higher source e.g. the help desk or other authorities then we will continue to suffer from the side effects of well-intentioned programmers. Remember that if you include a policy statement that limits the end user to allowing only active content from a trusted source to execute, then be prepared to list the "trusted sources" that the organization recognizes.

If every end user would just remember the one simple rule our parents used when we were young then many of the issues involving active content could be avoided. What is this golden rule that has saved so many youths from a life interrupted? And I quote "Don't take candy from strangers." Why, because the giver expects something in return and it may just be sinister indeed. This is no different then someone embedding a command within a macro or hidden within an HTML page or attached to your E-mail, and then freely giving it to you for a subsequent surprise. Unless you know the source and can trust the source don't accept their 'cyber candy' as it may contain things that will interrupt your life in unpleasant ways.

Active Content Types

Active content comes in many types these days from application macros, to applets to background scripts. All of these have potential weaknesses which can be exploited not because the programmers had that intention when they were created but because a few immutable principles that still plague us today, biological issues aside, these are.

People basically trust each other and cling to the arcane notion that it can't happen here. Attackers use this to their advantage when launching active content exploits; in fact they count on it. In the real world it is easy to recognize a threat such as a dark street or alleyway. When walking or driving through a bad part of town a person knows inherently that they should lock their doors, pick up their pace and walk well to the street side of an alley. Unfortunately these visual clues do not present themselves when one is cruising the streets of the "Information Super Highway". Because they plug into the Internet from the safety and comfort of their home they are lulled into a false sense of security. It needs to be driven home that the Internet has no boundaries, good side of town or bad side of town it is all one community. There are no visual clues save for the ones presented by dialogue boxes that something may be amiss and it is these warnings that must be read and heeded if one is to safely walk in Cyber Space.

Postscript®, Java™, JavaScript™ and Visual Basic® Script, ActiveX®, Macros, and Browser Plugins are all capable of being exploited through well-defined and documented active content features. One flaw of Postscript was mentioned above however it is important to remember that this is not the only printer definition language, which suffers from exploitable features. HP® has several vulnerabilities in their page description language and Ethernet attached hardware that can lead to a denial of service through the automatic execution of commands on the target device.

Browser Plugins

Browsers such as Netscape and IE are wonderful tools that provide an avenue for ease of use and consistent look and feel that were not available in the not so distant past. However these interfaces to modern commerce and information access cannot provide for every possible use that may be required by an end user. Plugins as noted earlier are not active content themselves but merely an enabler. They provide for things such as the animation of cursors up to complex three dimensional or live video and audio streams to occur on a users workstation.

Plugins come from a variety of resources like RealPlayer®, Shockwave®, and Microsoft® to name a few. While these are usually signed by using a digital certificate this only tells a user that the code manufacturer has been certified by the issuer to be a real company. The digital certificate in no way guarantees that the signed code is not designed to compromise a system and the issuer of the certificate takes no responsibility should a compromise occur.

The plugins themselves need to come from trusted sources and probably should be installed by administrators when rolling out new systems. It may also be a good idea to keep the ones that are required for users to do their jobs on an internal server where they have been tested and verified free of undesired additive code.

The content, which users download, needs to be filtered for harmful Trojans or other malicious content that might otherwise cause the release, destruction or integrity of information. Most industry security experts agree that the best way to do this is at the perimeter of the network with an adjunctive system to a firewall that acts as either a proxy or connects through a firewalls API services like those specified in the Open Platform for Secure Enterprise Connectivity (OPSEC). In such a way all content, in and out is checked for malicious content. If your really paranoid, and you should be, then implement Anti Virus software on the desktop as well. The defense in depth model as presented by SANS, Eric Cole, is one that more organizations should adhere to if they want to provide a secure computing environment.

Remember if your going to scan and filter content there are legal implications. For one your policy statements must clearly define that corporate E-mail will be scanned and filtered. For another this rule may not apply if your corporate boundaries extend beyond the territories of the USA. Many states recognize a common law 'Right of Privacy' and the *Electronic Communications Privacy Act, 18 U.S.C. §§ 2510-2710, 18 U.S.C. § 2511(1)(a)*, prohibits the unauthorized interception of electronic communications (including E-mail). It is a good idea to check with your corporation's legal counsel before spending money on content filters that may not be able to be implemented due to legal constraints.

Macros

Macros are wonderful inventions that allow a user to store keystrokes for repetitive tasks and can do anything from inserting boilerplate text to calculating the gravitational force of a sub nuclear particle.

While no one argues their usefulness it is important to remember that the Melissa virus is an example of a malicious macro written in Visual Basic® that used the hosts email address book and client software to propagate itself and deluge mail systems the world over. For those wishing to digest the anatomy of this attack Microsoft® provides an in depth white paper on their web site at URL:
http://support.microsoft.com/support/exchange/content/whitepapers/melissa.doc

Macro languages also include higher level programming constructs such as conditional processing and branching. Some also include the capability to execute external programs, which includes things like access to destructive applications like DELTREE.EXE or maybe something the attacker copied down to the target system, such as a back door.

It is a good idea to know what and when macros execute by setting security levels within capable programs to ask before they are executed. If your administrators have time with the six million other things they have to do, it is probably a good idea that macros from outside sources be tested before they are released to users.

Microsoft® now includes the capability to sign macros and it would be wise to evaluate all macros generated from external sources and then sign them if they are free from harmful side effects. The signing of macros should be done by an internal trusted source appointed by the security officer. In this way it is a simple task to guide the user on the safe execution of macros by simply stating that no macros signed by anyone other then the trusted internal source will be executed on corporate information systems.

ActiveX®

ActiveX® is part of the Microsoft® Component Object Model, COM for short, that provides reusable code segments for application programs. These compact code modules help control many aspects of Wintel applications and hardware, and allow for the automation of many background tasks.

The security model, such that it is, relies heavily on user interaction in order to ensure that unsigned controls are not downloaded or executed on a host system. Other than this very scary trust model ActiveX® controls have little restrictions on what they can do once they are given permission to proceed. Of the thousand or so registered controls only 50 to 100 have the marked designation as safe for scripting.

The problem with ActiveX® controls is they are vulnerable to exploits launched by text imbedded within HTML documents. Smith, Richard M of MIT has listed several such vulnerabilities that exist between the use of ActiveX® and HTML using trusted execution privileges and leveraging PC manufacturers (a trusted vendor) controls that ship with their systems.

Also remember that by default Wintel operating systems and applications execute signed or unsigned ActiveX® controls with great impunity and is one of those standard defaults that needs to be changed in order to ensure a safe computing environment.

JavaScript™ and Visual Basic® Script

Both of these languages belong to a class of scripting tools whose code can be embedded into Web pages for creating highly interactive documents. I won't go into the long debate on whose is more open and which is more powerful because this is irrelevant.

While the theory of their operation dictates that they cannot directly access a client file system and communications is restricted to the host from which the content originates there are numerous design and implementation bugs in both of these products.

The biggest flaw is that they work within the context of the browser and in theory are limited or bounded by whatever is legal within the browser. Unfortunately modern browsers are bound tightly to other applications such as email and have various plugins and ActiveX® controls that can be accessed. Additionally these scripts will assume whatever your privileges are on the systems you are logged into at the time of their execution. If you hit a malicious script while logged in as an administrator then the script will run with administrative privileges which could be devastating to security.

According to the NIST ITL Bulletin of March 2000 past exploits include stealing mail addresses, forging email, and passing your browsing habits to remote servers. CERT lists some even more dangerous possibilities in their February 2000 advisory at URL:

http://www.cert.org/advisories/CA-2000-02.html

Again it is important to remember that setting the browsers controls to limit the automaticity of script execution will help to protect the organization from an accidental breach of confidentiality or destruction that might occur.

Java™

Java™ is the universal code that is similar in nature to ActiveX® in stated purpose only? It is the reusable code set that can be written once and run on many different types of hardware platforms if there is an interpreter called the Java Virtual Machine (JVM) on the target host.

The resulting byte code created from Java™ is conveyed by an HTML web page as an applet. Java™ unlike ActiveX® tried to address the security shortcomings of other modular programming languages by addressing security from the onset. It provides a security architecture known as a sandbox. According to NIST the sandbox is a bounded area that "restricts the access of the code to computational resources based on its permissions." The problem with permissions is that they are defined again as trusted resources e.g. where it comes from and who wrote the applet.

While the sandbox is supposed to prevent the applet from accessing files or even changing them and prevent accessing the network, there have been many successful exploits that circumvent the sandbox security construct.

Brumleve, Dan proved this in August of 2000 when he listed a JVM applet on Bugtraq, that allowed the copying and execution of Brown Orifice, an NT back door exploit, using HTTPD as the transport. Other exploits include email forging, denial-of-service and those found in the JavaScript™ language.

These exploits can be particularly devastating if not checked at the perimeter of the network and end users blindly allow applets to download and run unabated. Remember to filter content and provide guidance in order to minimize the likelihood of an attackers success.

Rules to Lower Your Risk:

1. To paraphrase so many "Trust nothing, paranoia can be a good friend."
2. Know your systems, both hardware and software.
3. Strong policies and procedures are needed to ensure that active content is controlled at the human factor level.
4. Ongoing education and awareness is needed stressing the importance of limiting automatic execution of code from untrusted sources.
5. To perform administrative tasks elevate your privileges as required and immediately when finished return to unprivileged status.

6. Never browse to a web site while logged in with privileged status.
7. Any code installed on workstation's accessing corporate information systems needs to be tested to ensure that it does what is intended.
8. Content passing through corporate networks needs to be filtered on ingress and egress to minimize the possibility of acquiring or delivering malicious content.
9. Implement a defense in depth security model to provide the best likelihood of prevention or detection.
10. Set E-mail, applications and browsers to as high a security mode as is practical given the corporate culture and sensitivity of the information you are trying to protect.
11. Apply security patches to affected systems after testing them to ensure they don't break more then they fix.
12. Turn on as much logging as is practical on your systems to provide a safety net for detection and recovery from a compromise.
13. Make sure you have adequate backups and that you test them on a regular basis.

Trends

It is certain that we have only seen the tip of the iceberg in active content exploits. E-mail is undoubtedly the killer application of the e-enabled enterprise; it is also the most exploited venue for attacking information and its resources. It is easy to predict that these attacks will continue to rise until users understand the implication of their answers and organizations filter at the perimeter to reduce the likelihood of injurious payloads reaching their targets.

Looking into the crystal ball and divining the future is not necessarily the easiest thing to do. However as it relates to active content you can count on several trends to continue that will act as fuel for the attackers vehicles, these are:

1. The Internet will continue to grow in popularity.
2. Application programs will continue to grow in size and complexity.
3. Competitive pressures to bring products to market will undermine the thorough testing of applications.

When you put these three things together you will create a volatile mixture enabling an attacker with all they need to compromise your information. It is therefore the duty of all parties concerned with information security that they remain vigilant if they would protect the assets entrusted to them.

Suggested Areas of Further Study

Perimeter Defense Systems
Content Filtering and Legal Implications
Making Software More Secure
Policy to Lower Active Content Risks
Content Filtering Pros and Cons
Sandbox Security Does It Work

References

[1] Lane, Tad. "Active Content and Web Browser Security." August 1997. URL: http://www.lanl.gov/projects/ia/library/bits/bits0897.html

[2] Stewart, John N. "Active Content." WebServer Magazine February 1996. (1996): 20 – 22

[3] Festa, Paul. "A Question of Safety." 19 February 1998. URL: http://news.cnet.com/news/0,10000,0-1003-201-326605-0,00.html

[4] Farrow, Rick. "Dangerous E-Mail: Return To Sender." Network Magazine September 1999 (1999): 99 – 103

[5] Pavelec, Borris. "Automatically Disabling Active Content in E-Mail." 16 May 2000, URL: http://members.cotse.com/mailing- lists/ntbugtraq/2000/May/0069.html

[6] Microsoft. "How to Disable Active Content In Internet Explorer." Q154036. 9 March 2001. URL: http://support.microsoft.com/support/kb/articles/Q154/0/36.asp

[7] Jansen, Wayne & Karygiannis. "ITL Bulletin – Security Implications of Active Content." March 2000. URL: http://www.nist.gov/itl/lab/bulletns/mar00.htm

[8] Smith, Richard M. "Security Issues in HTML – Based E-mail." 7 February 2000. URL: http://www.pdos.lcs.mit.edu/asrg/02-07-2000.html

[9] Bramleve, Dan. "Why I Distrust Java." 4 August 2000. URL: http://cm.bell-labs.com/who/ehg/blindly.html

[10] Examples of misbehaved controls and applets – Use Caution[1] Running With Scissors

[1] Go to this web site then click on Online Security Test

Questions:

1. For active content exploits the attacker relies on?
    a. A fast Internet connection.
    b. Solid and secure programming code.
    c. An implied ethical standard.
    d. The Human Factor.

    The correct answer is D. Most active content exploits occur through the accomplice, usually an end user, ignoring a dialogue box that appears to warn them of the potential danger of their action.

2. Browser Plugins are a form of active content? True or False.

    False. Browser Plugins are an enabler, which can allow active content to be executed on the users local system.

3. Early printer exploits using the Postscript printer language were accomplished by?
    a. Elaborate and difficult to write scripts.
    b. Downloading a password to the printers NVRAM.
    c. E-mailing a file to the printer.
    d. Downloading a Trojan to the printers NVRAM.

    The correct answer is B. Early exploits of Postscript involved downloading a password known only to the attacker into the printers NVRAM. Subsequent print jobs that didn't know the password would not print.

4. Dialogue boxes are?
    a. Evil and should be outlawed.
    b. A means to carry on a conversation with a programmer.
    c. Warnings that require reading and interpretation.
    d. Prevent users from doing their work.

    The correct answer is C. Dialogue boxes occur to notify the user of an event that either has occurred or will occur if they so choose.

5. Most states recognize a common law 'Right to Privacy'? True or False.

    True. Almost all states recognize the rights of an individual to privacy.

6. One reason active content exploits are so successful is?
    a. They are very well written by people who have intricate knowledge of computer hardware and software.
    b. Because visual clues are limited and largely ignored.
    c. People have a real and well-earned sense of trust and security when it comes to applications.
    d. Because their mother never told them about the risks.

    The correct answer is B. People have a false sense of security when executing applications that leads them to largely ignore or misinterpret warnings that appear on their computer monitors.

7. JavaScript is more secure then Visual Basic Script? True or False.

   False. Both languages have serious flaws when it comes to security, to say one is more secure then the other would be akin to saying a rifle is more dangerous then a bow and arrow.

8. One of the most successful macro viruses ever written was the Melissa Virus? True or False.

   True. This virus illustrated clearly the weaknesses inherent to the active content structure and crippled mail systems the world over.

9. ActiveX was created with the thought of security first and functionality second? True or False.

   False. Most active content constructs had little if any forethought for security.

10. JAVA uses a security architecture referred to as?

    a. The sandbox.
    b. The playpen.
    c. The honey pot.
    d. Columbia's revenge.

    The correct answer is A. JAVA uses a security architecture known as the sandbox, which is to prevent the application from accessing the network or changing files on a target system.