ANALYZING THE ACTIVEX CONTROL ON IE7 VS. IE8

A thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science in Computer Science

By

Mani Heravi

August 2012
The thesis of Mani Heravi is approved:

_________________________________________  ______________________
Robert McIlhenny, Ph.D.  Date

_________________________________________  ______________________
John Noga, Ph.D.  Date

_________________________________________  ______________________
Jeff Wiegley, Ph.D. Chair  Date

California State University, Northridge
# TABLE OF CONTENT

The thesis of Mani Heravi is approved: ................................................................. ii

LIST OF FIGURES ............................................................... v

ACRONYMS .................................................................................. vi

ABSTRACT ...................................................................................... vii

CHAPTER 1: INTRODUCTION .................................................................. 1

  History .......................................................................................... 1
  Birth Of Internet Explorer .............................................................. 1
  ActiveX Control ........................................................................... 2
  Security Threats & Methods ............................................................ 5
  Browser Helper Object (BHO) .......................................................... 6
  Cross-Site Scripting Attacks ............................................................ 7
  Buffer Overflow ........................................................................... 9
  Stack Overflow ............................................................................ 10

CHAPTER 2: INTERNET EXPLORER 7 .................................................. 11

  Security Features ......................................................................... 12
  Protected Mode .......................................................................... 12
  ActiveX Opt-In ........................................................................... 14
  Phishing Filter ............................................................................ 15
  Group Policy .............................................................................. 16
  Internet Explorer Administration Kit 7 .......................................... 17

CHAPTER 3: INTERNET EXPLORER 8 .................................................. 18

  Security Features ......................................................................... 19
  DEP / NX Memory Protection ....................................................... 19
  Improvement In ActiveX Control ................................................... 20
  SmartScreen Filter ...................................................................... 21
  XSS Filter .................................................................................. 24
  ClickJacking Defense .................................................................. 25

CHAPTER 4: COMPARISON ................................................................. 27

  Improvement In Security Features ............................................... 28
  Improvement In Memory Management ......................................... 28
Improvement In Network................................................................. 29
Improvement In HTML Sanitization.................................................. 29
Improvement In MIME-Handling & File Uploading.......................... 29
REFERENCES .................................................................................. 31
APPENDIX A .................................................................................. 36
LIST OF FIGURES

Figure 1.3.1 – ActiveX Stand-Alone Scenario ......................................................... 3
Figure 1.3.2 – ActiveX Stored on a Land Scenario ................................................. 4
Figure 1.3.3 – ActiveX Stored on Web Server Scenario ......................................... 4
Figure 1.4.1 – Browser Helper Object .................................................................... 7
Figure 1.5.1 – Buffer Overflow .............................................................................. 9
Figure 1.5.2 – Stack Overflow ............................................................................... 10
Figure 2 – IE7 Process Model, Protected Mode ON/OFF ....................................... 11
Figure 2.1 – IE7 Protected Mode ........................................................................... 14
Figure 2.3.1 – Phishing Filter Detect Website ....................................................... 15
Figure 2.3.2 – Phishing Filter Validated Website .................................................... 16
Figure 2.4.1 – Active Scripting ............................................................................. 16
Figure 2.4.2 – Active Scripting ............................................................................. 17
Figure 3 – IE8 Process Model .............................................................................. 18
Figure 3.1 – IE8 Informational Box ....................................................................... 21
Figure 3.2 – SmartScreen® Filter User Interface ..................................................... 22
Figure 3.3 – Unsafe Download ............................................................................. 23
Figure 3.4 – IE8 XSS Filter ................................................................................... 24
Figure 3.5 – IE8 ClickJacking .............................................................................. 25
Figure 4.1 – IE7 HTML (Left) vs. IE8 Plain Text (Right) ......................................... 30
Figure A1 – Windows Rings .................................................................................. 36
Figure A2 – IE8 Add-Ons Management Interface ................................................. 37
Figure A3 – Process Explorer by SysInternals ...................................................... 38
Figure A4 – IE8 Error Message After Crash .......................................................... 38
Figure A5 – ASLR Before Reboot ........................................................................ 40
Figure A6 – ASLR After Reboot .......................................................................... 40
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASLR</td>
<td>Address Space Layout Randomization</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>DEP</td>
<td>Data Execution Prevention</td>
</tr>
<tr>
<td>NX</td>
<td>No Execute</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>XD</td>
<td>Execute Disable</td>
</tr>
<tr>
<td>XSS</td>
<td>Cross-Side Scripting</td>
</tr>
<tr>
<td>IE</td>
<td>Internet Explorer</td>
</tr>
<tr>
<td>CLSID</td>
<td>Class Identification</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>COM</td>
<td>Component Object Model</td>
</tr>
<tr>
<td>SDL</td>
<td>Security Design Lifecycle</td>
</tr>
<tr>
<td>SP</td>
<td>Service Pack</td>
</tr>
<tr>
<td>MFC</td>
<td>Microsoft Foundation Classes</td>
</tr>
<tr>
<td>ATL</td>
<td>Active Template Library</td>
</tr>
<tr>
<td>DDE</td>
<td>Dynamic Data Exchange</td>
</tr>
<tr>
<td>OLE</td>
<td>Object Linking and Embedding</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>UIPI</td>
<td>User Interface Privilege Isolation</td>
</tr>
<tr>
<td>UAC</td>
<td>User Account Control</td>
</tr>
<tr>
<td>LCIE</td>
<td>Loosely Coupled Internet Explorer</td>
</tr>
<tr>
<td>CSRF</td>
<td>Cross Side Request Forgery</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
</tbody>
</table>
ABSTRACT

ANALYZING THE ACTIVEX CONTROL ON IE7 VS. IE8

by

Mani Heravi

Master of Science

in Computer Science

Cross-site scripting vulnerabilities date back to 1996 during the early days of the World Wide Web. A time when e-commerce began to take off, the bubble days of Netscape, Yahoo, and the obnoxious blink tag. The Web is the playground of 800 million cyberecitizens, home to 100 million Web sites, and transporter of billions of dollars daily [3]. Economies have become dependent on the World Wide Web as a global phenomenon. Crackers found that when unsuspecting users visited their Web pages they could forcibly load any Web site (bank, auction, store, Web mail, and so on) into an HTML Frame within the same browser window. Then using JavaScript, they could cross the boundary between the two Web sites, and read from one frame into the other. Moreover Crackers also found out that by using IFRAME and redirection using JavaScript they can install keylogger or Trojan software on the victim’s machine in order to steal his/her personal data. Recognizing these challenges, I would like to examine some of the weaknesses and strengths of the Internet Explorer’s ActiveX Control on two
different versions of Internet Explorer and why ActiveX Control on IE8 can be better than ActiveX Control on IE7.
CHAPTER 1: INTRODUCTION

History

In 1995, Microsoft released add-on software for their Windows 95 Operating System called Internet Explorer also known as Microsoft Internet Explorer, which is usually abbreviated as IE. IE is used for graphically surfing the Internet. Thomas Reardon starting in summer 1994 created Internet Explorer. Starting from Internet Explorer 3, Microsoft’s web browser was sophisticated enough to compete with the reigning browser of the time Netscape.

Internet Explorer is one of the most used applications on desktop computers. Even though this browser doesn't produce any business documents such as Microsoft Word or Excel, it is still the most used application among consumers.

Birth Of Internet Explorer

Internet Explorer 7 (IE7) released in October 2006 by Microsoft was a major update to the previous version of Internet Explorer (IE6). After five years of development it was shipped as the default browser for Windows Vista and Windows Server 2008. It was also offered as an upgrade to the previous version, IE6, on Windows XP. Starting with this release of Internet Explorer, Microsoft stopped supporting Windows 2000 and below. They introduced new features with IE7 to increase their security on their browser such as:

- Internet Explorer Protected Mode
- ActiveX Opt-in
- Cross-domain Scripting Attack Protection
- Security Status Bar
- Phishing Filter
• Parental Control

Some of the above security features are only available in Windows Vista and above, such as Protection Mode and Parental Control that enables a safer environment for a user’s Internet browsing.

On March 19, 2009 Microsoft announced the release of a new version of their browser, Internet Explorer 8 (IE8). This new release was only for Windows XP and above. Two years later, Microsoft released their latest web browser in March 2011 called Internet Explorer 9, IE9, as an update to Windows Vista, and Windows 7. New features for Internet Explorer 8 include:

• Safety Filter
• Greater Control Over ActiveX Control
• New AJAX Feature for Safer Mashups (XDomainRequest, and XDM)

Since most of people in the world use Microsoft Windows, Internet Explorer is the most popular browser among consumers. As of February 2012 Internet Explorer 8 holds 14.71 percent of the market share while Microsoft Internet Explorer holds 30.7 percent of market share. This makes IE a common browser to target for malicious websites.

**ActiveX Control**

Before we jump into why ActiveX can be a security risk to users and how Microsoft minimizes those threats in Internet Explorer 7 and 8, we need to understand what ActiveX is and how it works.

ActiveX, also known as ActiveX Control, is an OLE object (Object Linking and Embedding Object). ActiveX is derived from two different components a Dynamic Data
Exchange (DDE), and Object Linking and Embedding (OLE) [14]. It provides a reusable framework to end-users. Due to the difficulty of OLE and lack of support on Microsoft Foundation Classes (MFC), Microsoft reintroduced ActiveX Controls in 1996 with a simple and easy to use specification. ActiveX Controls use an OCX extension that was inherented from OLE. This file extension is a second generation of component architecture [13]. The first generation was VBX since they were written in Visual Basic. These controls are run on behalf of the client program such as 3D toolbars, notepad, Microsoft Excel. These controls can be maliciously provided to the client's browser if the user navigates to a controlled website, which turns the client's browser to be controlled by the malicious server and provide unwanted access to the client side.

These controls can be written in Microsoft Foundation Classes (MFC), Active Template Library (ATL), C++, or any other language with support for Component Object Model (COM). There are mainly three different ways to use ActiveX Controls [13]:

- Stand-Alone scenario is where the ActiveX Control resides on the computer and is executed from a Host Program on that machine (Figure 1.3.1). Adobe Flash Player for Internet Explorer is an example of this scenario.

  ![ActiveX Stand-Alone Scenario](image)

  *Figure 1.3.1 – ActiveX Stand-Alone Scenario*

- Stored on a LAN (Local Area Network), Run Locally scenario is if the ActiveX control is not installed on the local machine, The Host Program must
request it for download from a LAN server or SMB and then execute it locally (Figure 1.3.2).

Figure 1.3.2 – ActiveX Stored on a LAN Scenario

- Stored on a Web Server, Run Locally is the most common scenario where a user visits a webpage that links to an ActiveX Control. If the browser doesn’t already have that control installed, the user is notified. If the user trusts the page, they can install that control from the web server (Figure 1.3.3). That control is then executed locally after it is installed on user’s machine.

Figure 1.3.3 – ActiveX Stored on Web Server Scenario
A Web developer can link their webpage to ActiveX Controls in multiple ways, but the easiest way is to use the OBJECT tag in HTML with several different attributes. These Attributes specify control parameters. The code below is a sample HTML code that links an ActiveX Control to the webpage.

<OBJECT
    ID="MyActiveX"
    WIDTH=350
    HEIGHT=50
    CLASSID="CLSID:36299202-09EF-4ABF-ADB9-47C599DBE778">
  <PARAM NAME="_Version" VALUE="65536">
  <PARAM NAME="_ExtentX" VALUE="2646">
  <PARAM NAME="_ExtentY" VALUE="1323">
  <PARAM NAME="_StockProps" VALUE="0">
</OBJECT>

Starting with Internet Explorer 5 Microsoft added support for XMLHttpRequest for the MSXML ActiveX module that made their web browser more unsecure and vulnerable to attacks while planting the seed for a revolution that would become Web 2.0.

**Security Threats & Methods**

As consumer demand for online services grows, web applications will run a wide range of services on the client side using ActiveX Controls. Each of these services requires users to provide sensitive information such as online banking information, which, from an IT security stand point, is a security risk since ActiveX Controls allow execution of arbitrary code in the browser. Most IT Security and System Administrators block all ActiveX Controls for this reason [22].
After the release of Internet Explorer 7, Symantec Internet Security Report observed the different attacks on different browser plug-ins. 79 to 89 percent of attacks addressed vulnerable ActiveX Controls with far less attacks occurring in Java or QuickTime plug-ins due to the popularity of Internet Explorer [31].

**Browser Helper Object (BHO)**

Internet Explorer, like any other Win32 applications has its own memory space. Internet Explorer can execute the process of Browser Helper Object at start time and such a process can share a memory space with Internet Explorer and perform any actions that are available on Window or modules such as GoBack, DocumentComplete, or access browser’s toolbar [27]. BHOs are based on COM Object (Figure 1.4.1) and each has a CLSID. If registered under a certain registry entry, Internet Explorer executes them for each window tied to browser’s main window. Most Malware uses this feature of Internet Explorer by installing a malicious BHO that tricks users into installing software such as fake anti-virus by showing informational popup messages or fake advertising while they are surfing the net using Internet Explorer.
Microsoft released a security advisory to warn developers on some of the features in BHO such as \textit{IHTMLDocument2::domain} where if used without caution, can lead to an attack such as Domain Name System (DNS) manipulation [28]. Such a feature allows Cross-Domain Scripting on the server.

\textbf{Cross-Site Scripting Attacks}

Web application's flaws have lead to a rapid growth of Cross-site Scripting Attacks (XSS) within the past few years. An attacker can use embedded Javascript to send malicious content to the web server. Websites like \textit{xssed.com} have collected thousands of XSS vulnerabilities that can be used on webpages across the net. These XSS vulnerabilities enable attackers to gain control over the relationship between client and web server to facilitate Cookie-Theft, Session Hi-Jacking, and Keystroke Logging and more importantly perform actions on the web server on behalf of the user [12]. There are mainly three different levels or categories of XSS attacks, reflected-XSS, stored-XSS, and DOM base XSS [23, 24].
• Reflected-XSS Attack:

This is the most common way of Cross-site Scripting Attack. With this attack, the injected code is reflected from web server and delivered to user’s browser for example by email messages [24]. The user is tricked into clicking on the link that contains malicious content. Reflected-XSS attack can be easy turn into DOM Based-XSS attack.

• Stored-XSS Attack:

Stored-XSS attack is the same as Reflected-XSS attack but with a minor difference. With this attack the injected code permanently stays on the web server. This is due to not validating user input from web server either by forms, database, or by URL queries [24].

• DOM Based-XSS Attack:

When the payload of the attack is executed as a result of modification in Document Object Mode (DOM) tree, it is called a DOM Based-XSS attack. This attack manipulates the webpage content and dynamically generates computed content using Javascript. A typical example is extracting data from URL via location DOM object.

Recently Cross-site Scripting Attack added a new vector, which addresses the vulnerable plug-ins on the client web browsers such as ActiveX Controls or plug-ins written in Java using either heap or stack overflow methods.
Buffer Overflow

Buffer Overflow attacks happen when an attacker’s malicious code attempts to inject large amounts of data into a program’s buffer, which is a set of sequential memory blocks that the operating system has assigned to the program. As a result, an attacker gains control of the next instruction that must be executed and forces it to execute his/her program instead. Another side effect of buffer overflow is to crash the current process or operating system.

```c
#include <stdio.h>
void cp(char *d);

int main(int argc, char ** argv) {
    cp(argv[1]); /* argv[0] is the name of program */
    return 0;
}

void cp(char *d) {
    char buffer[10];
    strcpy(buffer, d); /* Buffer Overflow */
}
```

Figure 1.5.1 – Buffer Overflow

Figure 1.5.1 shows that `buffer` variable expected a string with a size of 10 bytes. Anything more than 10 bytes lead the above source code to a buffer Overflow.

Another example of buffer overflow is when websites such as Facebook or MySpace use ActiveX Controls for different tasks such as image uploading. In January 2008 Elazar Broad discovered a buffer overflow in MySpace Image Uploader, which allowed an attacker to gain access to the victim’s computer remotely [29, 30]. This ActiveX Control vulnerability was due to improperly checking the boundary of the string that gets set to Property string.
The size of the string in the property value field is limited to 250 characters and by overlaying long strings causes a stack base buffer overflow and leads to execution of arbitrary code on a victim’s machine.

**Stack Overflow**

Stack Overflow Attacks are another vector of buffer overflow attack where the buffer that is written by malicious code resides in the operating system stack. Figure 1.5.2 demonstrates a stack overflow when the input to the program is larger than buffer size.

```c
#include <stdio.h>
#define BUFSIZE   10

int main(int argc, char ** argv) {
    char buffer[BUFSIZE];
    strcpy(buffer, argv[1]);
    return 0;
}
```

*Figure 1.5.2 – Stack Overflow*
CHAPTER 2: INTERNET EXPLORER 7

Before talking about Internet Explorer 7 security features that protect users from browser and browser plug-in vulnerabilities, we need to know how all IE7 components work together by explaining its process model (Figure 2).

![Figure 2](image)

**Figure 2 – IE7 Process Model, Protected Mode ON/OFF**

Figure 2 shows the simplified version of the IE7 Process Model [32]. Each browser’s window in Internet Explorer 7 runs in the same process meaning that if a user presses CTRL-N to open a new IE7 process, that UI Frame runs in the same process as other UI Frame and shares Toolbar Extension, Browser Helper Objects and more importantly ActiveX Controls. This is a security risk because one vulnerable ActiveX Control, let’s
say a buffer overflow, can lead to compromising all the other windows. This can result in leaking users’ sensitive information such as username and password, or the attacker can gain remote access to the victim’s machine and lunch another attack on behalf of user without his/her knowledge.

**Security Features**

Internet Explorer 7 built on the security improvements in Windows XP service pack 2 (SP2). Security enhancements for Windows XP that was focused on Internet Explorer includes popup blocker and tightening Internet Explorer's behavior through its settings and to help users to gain more control over their browser.

**Protected Mode**

This new feature of IE7 was only available in Windows Vista and was a big improvement from Microsoft to secure Internet Explorer. Protected mode protects users from execution of unauthorized controls in the Internet Explorer process and minimizes the risk by restricting process access level (Please refer to Appendix A, Windows Internals). This feature uses built-in security mechanisms in Windows Operating System such as [18]:

- User Account Control (UAC) escalates the user to a higher access level if they are running with limited access and the processes needs administrative privileges to either install or write into system object.
- Integrity Mechanism to restrict write access from processes with lower access levels.
- User Interface Privilege Isolation (UIPI) blocks processes from sending certain windows messages other than User APIs.

In order for an attacker to gain access to a victim's machine they must use an exploit that uses a vulnerable control with high access level (Please refer to Appendix A, Windows Internal) and gain write permission to the victim's machine. Protected Mode is built on
new features of Windows Vista called Integrity Mechanism (Please refer to Appendix A, Windows Internal) to block such an attack from running in Internet Explorer 7 on such a low access level and restrict write access to system objects such as processes, controls, and registry keys with higher access level. Low access level processes can only write into system objects with the same access level, low. As a result, in Protected Mode, Internet Explorer 7 can only write to system objects with low access level such as Internet Temporary Files folder, History folder, Cookies folder, Favorite folder and Windows Temporary folder %TEMP% [18]. This feature can be configured via Group Policy.

Moreover, Protected Mode can only send certain windows messages to system objects with higher access level by sandboxing the browser’s services using Windows Vista UIPI, which prevent the attacker from gaining full control over the victim’s computer [18]. UIPI prevents processes with lower access level to use processes with higher access level by blocking these behaviors [19]:

- Window handle validation for processes with higher access level
- Blocks SendMessage(), PostMessage() API function call
- Thread Hooks being used to Access Processes with Higher Access Level
- Journal Hooks being used to Monitor Processes with Higher Access Level
- Prevent DLL Injection to Processes with Higher Access Level
As shown in figure 2.1, compatibility layer uses Windows Compatibility Shim to handle the need of the old extensions or controls by intercepting the write call to the Document folder under user profile and registry key HKEY_CURRENT_USER [18]. Two broker processes, User Privilege Broker (IEUser.exe) and Administrator Privilege Broker (IEInstal.exe) handle write access to processes with higher access levels and installing ActiveX Controls in Internet Explorer respectively.

By preventing unauthorized write access, Protected Mode limits the possibility of attacks and the amount of damages that an attacker can do on a victim's machine by either exploiting vulnerable ActiveX Controls or by poorly written web applications.

**ActiveX Opt-In**

Before Internet Explorer 7, administrators were forced to disable the ActiveX Controls from their users’ browsers to enhance their network security. Starting with Internet Explorer 7, Microsoft offered a new powerful feature called ActiveX Opt-in [15]. This
new feature disables all the ActiveX Controls that are not allowed by the user. If the website wants to install any ActiveX Control in the user’s browser, with this feature users will be notified by an Information Bar and the user can decide to Accept or Decline the installation.

By default ActiveX Opt-In only applies to controls that are used by Internet Explorer and restricted site zones while this feature does not affect those controls that are part of intranet or trusted site zone [21].

**Phishing Filter**

Starting with Internet Explorer 7, Microsoft offered a new feature called Phishing Filter to block and notify the user about spam. This is a dynamic security feature that uses *Extended Validation Certificates* (Please refer to Appendix A, Extended Validation Certificates) combined with its database that contains a list of known phishing websites to stop attackers from installing malicious controls or stealing users sensitive information.

![Phishing Filter Detect Website](image)

*Figure 2.3.1 – Phishing Filter Detect Website*

When Phishing Filter is enabled and the user visits a website that is marked as a phishing website in the IE database, the background of the address bar turns red color and the text *“Phishing Website”* (Figure 2.3.1) appears to inform user about the identity of the website.
Moreover, if IE suspects the visited website is a phishing website, the color of the address bar turns to yellow with the text “Suspicious Website”. Furthermore, if the identity of the website can be validated with a website SSL certification, the IE7 address bar turns to green with information on the SSL authority (Figure 2.3.2).

**Group Policy**

*Group Policy* can be used to limit certain features from Internet Explorer to minimize the threat. Administrators can protect their users from malicious programs by disabling active scripting in response to specific threats. A few years back, Microsoft created a technology called Security Design Lifecycle (SDL), to reduce the impact of software vulnerabilities. This technology is very effective because if a vulnerable ActiveX Control exists but there is no update countering the attack, we call this a zero day attack or after updating if that control is still vulnerable, administrators can block that Control by disabling "Allow Active Scripting" from Group Policy (Figure 2.4.1).

<table>
<thead>
<tr>
<th>Policy Object</th>
<th>Location</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Active Scripting</td>
<td>Computer Configuration\Administrative Templates\Windows Components\Internet Explorer\Internet Control Panel\Security Page&lt;ZONE&gt;</td>
<td>This should be Disabled to minimize the threat such as Zero Day Attack</td>
</tr>
</tbody>
</table>

*Figure 2.4.1 – Active Scripting*

By default, Internet Explorer allows scripts to open, reposition, and resize the window. This can allow popups to appear on the user’s computer. Popups are one vector of computer attacks that can lead to exposing a user’s sensitive information or
compromising their computer. Using a Group Policy, administrators can stop those scripts from running in Internet Explorer (Figure 2.4.2).

<table>
<thead>
<tr>
<th>Policy Object</th>
<th>Location</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scripted Window Security Restriction</td>
<td>Computer Configuration\Administrative Templates\Windows Components\Internet Explorer\Internet Control Panel\Security Page&lt;ZONE&gt;</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**Internet Explorer Administration Kit 7**

In the corporate world, managing the web browser becomes more critical as the demand on corporate intranet and extranet grows. This feature of Internet Explorer 7 offers improvement in productivity, manageability and reliability [17]. This feature allows administrators to create custom packages for IE7 on Windows XP and above to allow network administrators to establish several levels of control on the user [37].

- Customize the user’s Internet Explorer 7 homepage and feed.
- Customize IE7 to install specific toolbars.

With this feature, administrators can customize ActiveX Controls as well as Security Settings that is suitable for their organization.
CHAPTER 3: INTERNET EXPLORER 8

The previous chapter explained the process model of Internet Explorer 7 before covering its security features. Similarly, this chapter starts with describing Internet Explorer 8 Process Model (Figure 3), and then covers IE8 security features.

As we saw in the previous chapter, all IE7 windows are running in the same process, which is a potential security risk. Internet Explorer 8 addresses this issue by launching each tab in a separate process as well as Toolbar Extensions, Browser Helper Object and ActiveX Controls. UI Frame and Broker Object are in the same process [33]. This causes better launch time performance while Broker Object determines what mode IE is
configured to be lunched in. When in Protected Mode, IE must be launched on a separate level to maintain the integrity of the protected tab.

Moreover, the integrity mechanism operates on each process where each tab is running. With this new design each tab can have Protected Mode ON or OFF without changing other tabs settings.

**Security Features**

Besides all of the new features that are implemented and offered by Internet Explorer 7 such as ActiveX Opt-In, Internet Explorer 8 introduced a new security measurement that targets three major security issues: social engineering, web server security, and browser-based vulnerabilities [8].

**DEP / NX Memory Protection**

Internet Explorer 8 offers new features called Data Execution Prevention (DEP) or No-Execute (NX) to prevent the execution of arbitrary code on memory locations that are marked as non-executable [1]. This feature, combined with ASLR (Please refer to Appendix A, Address Space Layout Randomization), protects users from Buffer Overrun Attacks not only in the browser but also in browser plug-ins.

Windows XP SP3 does not support ASLP, but relies solely on DEP/NX. Still it’s a challenge for an attacker to execute malicious code in browsers or browser plug-ins in memory.

Starting with Windows XP SP2 and Windows Server 2003 SP1, Microsoft started to support DEP/NX on its x86 Operating System. DEP/NX is built on a feature of newer CPUs that allows marking memory pages as executable or non-executable (Please refer to
Appendix A, CPU Prevents Malicious Code Execution) [4]. DEP/NX works in a similar way but on the software level. For example, if DEP/NX sees a JUMP instruction that is not part of the process execution, it intentionally safely crashes that process to prevent the exploit from executing (Please refer to Appendix A, DEP/NX View and Crash).

Traditionally programmers used the _NXCOMPAT_ linker flag to enable this feature but Internet Explorer 8 introduced a new API function call _SetProcessDEPPolicy()_, which gives the programmers the following benefits:

- Users can disable DEP/NX as desired from either Group Policy or Internet Control Panel Checkbox.
- It enables DEP/NX on Windows XP SP3
- It ensures _ATL_THUNK_EMULATION_ to work properly.

**Improvement In ActiveX Control**

ActiveX Opt-In was introduced in IE7 Internet Explorer 8 offers new features that improve the security of ActiveX Controls such as Per-User (Non-Admin) ActiveX, Per-Site ActiveX, and Working with Users through managing their Add-Ons [8].

Starting with Windows Vista, Internet Explorer 8 introduced a new feature that forces ActiveX Controls to run with User (Non-Admin) privileges. This helps organizations to take advantage of User Account Control by allowing standard (non-admin) users to install ActiveX Controls. If the user somehow installs a malicious ActiveX Control, the overall system stays intact.
Figure 3.1 – IE8 Informational Box

When the user opens a website that contains an ActiveX Control, Internet Explorer 8 checks to see if the control has permission to be installed and run under the users account. This check is often called Per-Site ActiveX control check. Internet Explorer 8 inherited Informational box from IE7 (Figure 3.1) and if the control doesn’t have proper privileges, IE8 notifies the user about the attempt of the website to install an ActiveX Control.

Furthermore, Internet Explorer 8 allows users to find information about their installed ActiveX plug-ins through the Manage Add-Ons screen. Users can disable unwanted add-ons or plug-ins that are vulnerable. This is also an important feature for the plug-ins and add-ons developers to ensure that their product is secure and works as expected (Please refer to Appendix A, Internet Explorer Add-Ons Management).

SmartScreen Filter

This new feature is built on Phishing Filter, a successful feature from IE7, with a number of improvements to protect users from attackers [10].
• Friendlier User Interface
• Improved Performance and Faster Response Time
• Anti-Malware Support
• Better Support for Group Policy

This new feature is more efficiently uses computer resources as well with a lower impact on the browser’s speed. SmartScreen® filter runs parallel to the website being loaded. If the browser detects the website as being unsafe it shows a user-friendly error message to warn users (Figure 3.2)

Figure 3.2 – SmartScreen® Filter User Interface

This new feature goes beyond anti-phishing and helps to protect users from malware. Anti-Malware is URL-Repetition-Based [10]. This means that when the page is in the process of loading, this feature looks at the connected servers that are downloading page data in the browser and determine if those servers are known as a distributor of malicious content. Moreover, to expand the security, the data of the URL are transferred to SmartScreen web service for evaluation in encrypted form via HTTPS protocol [10].
Furthermore, if a user clicks on the direct link that downloads unsafe controls, this feature interrupts the download and warns the user about suspicious activities (Figure 3.3).

![Unsafe Download](image)

*Figure 3.3 – Unsafe Download*

As explained in the previous chapter, *Group Policy* is a core component for safe browsing. In IE8, *Group Policy* can be used to enable or disable SmartScreen filter for the entire Windows domain [10]. Administrators can use *Group policy* to prevent the users from modifying settings of this feature. Also, this prevents attackers from convincing the user to change or modify these setting and blocks further attacks.

SmartScreen also blocks malware. In order to understand how this feature prevents malware and protects user from it, we need to know what malware is. The authors of malware have always tried to infect user’s machine by tricking them into thinking that they are visiting or using a legitimate website or program [25].

With this feature, Microsoft and Internet Explorer developers have an easy job of blocking websites that contains malicious code such as malware by updating the SmartScreen database.
**XSS Filter**

This partnership is a collaborative between Security Software Engineers from SWI team and The Internet Explorer team has brought new features to IE with the hopes of making reflected-XSS attacks much more difficult to exploit on vulnerable environments [12].

XSS Filter monitors all incoming and outgoing messages through the web browser. If this filter discovers XSS attacks in a browser’s messages, it simply blocks them from execution and notifies the user. Figure 3.4 demonstrate such a behavior where the requested page has been modified by IE8 and the XSS attack is blocked.

![Figure 3.4 – IE8 XSS Filter](image)

XSS Filter must handle different scenarios of attacks and filters them appropriately.

Moreover, it must counter all different vectors of XSS attack that are not addressed by Surface Reduction. Here are some examples [12]:

- Filter must be effective to address and leverage adjusted attacks on web applications.
- Filter must not introduce new attack scenarios.
ClickJacking Defense

One of the most interesting vulnerabilities on web applications is Cross Site Request Forgery (CSRF) [33]. Preventing this sort of attack is very difficult because there is no easy fix out there. For example, a new Internet Explorer 8 object called XDomainRequest allows cross-domain request under some permissions from the server, which can lead to a CSRF attack. IE8 developers had to be sure with this feature they didn’t introduce new attacks [33]. Most web applications use Challenge Tokens to protect themselves from CSRF, but Challenge Tokens are subject to vulnerabilities. ClickJacking is a vector of CSRF where a user is tricked into clicking on hidden web elements, usually a hidden IFRAME, that leads to an unwanted transaction without the user’s knowledge.

Figure 3.5 – IE8 ClickJacking

A successful ClickJacking attack can bypass CSRF protection. When the user is tricked, attackers can steal sensitive information or use a vulnerable ActiveX control to do buffer or stack overflows to gain ring 0 (root) access to the victim’s machine by using specially crafted JavaScript with a malicious payload.
Internet Explorer 8 like other web browsers worked with web developers to mitigate this sort of attack by introducing an option in HTTP response header called \textit{X-FRAME-OPTIONS}. This option restricts the webpage on how it may be framed. If this option is set to \textsc{DENY}, IE8 prevents the page from framing (Figure 3.5) and if set to \textsc{SAMEORIGIN}, IE8 prevents the rendering of pages that are not part of the top-level domain of the webpage.
CHAPTER 4: COMPARISON

Internet Explorer 6 had lots of vulnerabilities not only in the browser’s default plug-ins but also in the browser itself. Today attackers are still finding security holes in IE6. Such vulnerabilities are published in journals like KillBits as their security report in December 2011 shows [36]. Microsoft introduced Internet Explorer 7, a successful successor browser from IE6 in terms of security by introducing new features such as ActiveX Opt-In, Protected Mode and Phishing Filter. But the design of Internet Explorer 7 wasn’t secure as all windows were running in a same process and sharing Toolbar Extensions, BHO, and ActiveX Controls. With Internet Explorer 8, Microsoft changed the process model of their browser to become more secure as well as adding new features to provide the user with situational awareness in browsing the web.

As we saw in the previous chapter, Internet Explorer 8 shipped with lots of features that make the browser and ActiveX Controls more secure such as DEP/NX memory protection, User Account Control (only available in Windows Vista) and improvements to Protected Mode from IE7.

Moreover, Starting with Internet Explorer 8, Microsoft started to work closely with other security teams to mitigate vulnerabilities in their browser and make Internet Explorer a safer browser. As an Example, Internet Explorer 8 supports other controls such as .NET, which work the same as ActiveX Controls and they run in a sandboxed environment such as .NET UserControl [7]. In summer 2008 at BlackHat conference Alexander Sotirov and Mark Dowd explained a way to bypass DEP/NX features in IE8 using .NET Controls such as .NET UserControl [6]. Attackers can use .NET framework DLLs to allocate executable memory in a predictable memory location in the Internet Explorer process.
This allows an attacker to bypass DEP/NX and exploit vulnerable ActiveX plug-ins. Microsoft responded with disabling .NET controls from running in the internet zone under default settings of medium or high and created a new control called URLAction that restricts the use of .NET UserControl.

Furthermore, Internet Explorer developers improved IE8 performance of their security features, memory management, network and data handling [35].

**Improvement In Security Features**

Internet Explorer 8 uses multi-threaded processing to enhance their security features. While the webpage is getting loaded, IE8 lunches different threads to analyze that webpage such as XSS Filter, SmartScreen Filter and ClickJacking defense. These improvements enhance the surfing experience for the user when they are using IE8 with speed and security.

**Improvement In Memory Management**

Microsoft invested heavily to improve the memory management in Internet Explorer 8. Memory leaks are potential security risks that can lead to buffer or stack overflow attacks. Microsoft fixed around 400 bugs that cause memory leaks and improved memory fragmentation of AJAX to reduce memory usage of webpages that have embedded AJAX by introducing *Garbage Collection* in IE8 [35]. This improvement also speeds up the launch time of IE8 over IE7.

Each window in Internet Explorer 8 runs in own process meaning that each is sandboxed. This improves the security while one window uses a vulnerable ActiveX control, information in other windows/tabs is still safe. Also, IE8 introduced DEP/NX Memory
Protection, which is enabled by default to foil attackers by preventing malicious code from being executed in the memory locations that are marked as Non-Executable areas [34]. This feature improves the security of vulnerable ActiveX controls.

**Improvement In Network**

One of the big advantages of IE8 over IE7 is the network improvement by increasing the number of download connections available. Internet Explorer 7 was limited to 2 connections per download whereas Internet Explorer 8 has overcome that limitation to 6. With this improvement the download time in IE8 is three times faster than the download time in IE7.

**Improvement In HTML Sanitization**

Internet Explorer 8 sanitizes HTML pages better than Internet Explorer 7 by introducing window$\text{Object}$ called $\text{toStaticHTML}$ [34]. When strings of HTML code passed into this function, any executable code that causes potential threat will be removed. This prevents most of the XSS attacks by sanitizing the input either from FORM or from URL queries.

**Improvement In MIME-Handling & File Uploading**

INPUT tag from HTML for uploading files to web server has lots of vulnerabilities such as attackers uploading malicious files to web servers by either stealing keystrokes or using a specially crafted code to attack the web server on behalf of the user. Internet Explorer 8 made this input a read-only box and forces the user to explicitly select the file to prevent this kind of attack [34].

*Picture Sharing* is another vector of attacks where an attacker embeds crafted code into images. IE8 blocks the script from image/* content type. The server declares that the file
is an image by setting X-Content-Type-Options of HTTP response header to nosniff [34]. This feature is also available in IE7 but IE7 renders the blocked image as an HTML while IE8 uses a more secure approach by loading the page in Plaintext (Figure 4.1).

![Figure 4.1 – IE7 HTML (Left) vs. IE8 Plain Text (Right)](image)

Internet Explorer 8 handles file downloading in a more secure way than Internet Explorer 7. By setting X-Download-Options of HTTP response header to noopen users are forced to download the file first before executing it [34]. In IE7 users can open and execute the file directly from the browser. This approach won’t allow attackers to download and execute malicious program on a victim’s browser and gain ring 0 (root) access (please refer to Appendix A, Windows Internal) over victim’s machine using vulnerable ActiveX Control without user’s knowledge.
REFERENCES

   April 8, 2008 11:00 AM
   Retrieved Date: 5 December 2011

   26 May 2006 4:24 PM
   Retrieved Date: 8 December 2011

   May 4, 2009
   Jones & Bartlett Publishers
   Retrieved Date: 11 December 2011

   10 Oct 2009 11:51 AM
   Retrieved Date: 16 December 2011

[5] Understanding DEP as a mitigation technology part 1 – Swiat
   12 Jun 2009 1:30 PM
   Retrieved Date: 18 December 2011

[6] Released build of Internet Explorer 8 blocks Dowd/Sotirov ASLR+DEP .NET bypass – Swiat
   23 Mar 2009 2:35 PM
   Retrieved Date: 21 December 2011

[7] DotNet UserControls Restricted in IE8 – Eric Law
   8 Oct 2009 10:17 PM
   Retrieved Date: 26 December 2011

[8] IE8 Security Part II: ActiveX Improvements – Matt Crowley

31
May 7, 2008 2:30 PM
Retrieved Date: 29 December 2011

[9] Add-on Management Improvements in Internet Explorer 8 – Christopher Vaughan
March 20, 2008 3:20 PM
Retrieved Date: 2 January 2012

[10] IE8 Security Part III: SmartScreen® Filter
July 2, 2008 12:00 PM
Retrieved Date: 5 January 2012

November 7, 2006 3:00 PM
Retrieved Date: 12 January 2012

July 2, 2008 12:03 PM
Retrieved Date: 14 January 2012

[13] Introduction to ActiveX - Part One
Nov 2007 2:27 AM
Retrieved Date: 17 January 2012

Retrieved Date: 19 January 2012

July 9, 2007
Retrieved Date: 22 January 2012

[16] Application Compatibility - IE7 Protected Mode
19 Apr 2007 11:44 PM
[26] IE8 Security Part IX - Anti-Malware protection with IE8’s SmartScreen Filter – Eric Lawrence
March 25, 2009 8:42 PM
Retrieved Date: 29 February 2012

[27] Browser Helper Objects: The Browser the Way You Want It – Dino Esposito
January 1999
Retrieved Date: 4 March 2012

[28] Security Considerations: Hosting and Reuse
April 9, 2004
Retrieved Date: 8 March 2012

Retrieved Date: 12 March 2012

Retrieved Date: 17 March 2012

April 2008
Retrieved Date: 21 March 2012

[32] IE8 and Loosely-Coupled IE (LCIE) – Andy Zeigler
March 11, 2008 6:50 PM
Retrieved Date: 23 March 2012

[33] IE8 Security Part VII: ClickJacking Defenses – Eric Lawrence
January 27, 2009 9:40 PM
Retrieved Date: 28 March 2012

[34] IE8 Security Part V: Comprehensive Protection – Eric Lawrence
July 2, 2008 12:05 PM
Retrieved Date: 31 March 2012

[35] IE8 Performance - Christian Stockwell
August 26, 2008 7:19 PM
Retrieved Date: 4 April 2012

[36] Assessing the risk of the December 2011 security updates
13 Dec 2011 10:14 AM
Retrieved Date: 7 April 2012

[37] IEAK 7 Beta 3 is now available – McoreD
July 21, 2006 5:30 PM
Retrieved Date: 10 April 2012

[38] Extended Validation Guidelines v1 Released! – Markellos Diorinos
June 12, 2007 9:57 PM
Retrieved Date: 14 April 2012
APPENDIX A

Windows Internals:

The Intel x86 family of CPUs use *rings* as a concept for access controls (Figure A1). There are four rings available starting from ring 0 where the OS Kernel reside with the most privileges and ring 3 with the least privileges where user applications are running. One way for viruses to gain access to ring 0 from ring 3 is by stealing a Token from other processes that are running with a higher access level. Rootkits are normally running the same level as OS kernel, in ring 0.

![Windows Rings Diagram](image)

*Figure A1 – Windows Rings*

The CPU enforces access levels on each process by memorizing each processes access level [3]. Running Internet Explorer in protected mode uses the same concept as the Intel x86 CPU family by separating the access level of processes that are running in Internet Explorer from LOW to HIGH.
Internet Explorer Add-Ons Management:

Microsoft offered Add-Ons Management for their browser starting with Windows XP SP2. Within the Add-On Management interface, users can view add-ons information by Name, Publisher, Version, and Class ID [8, 9] (Figure A2).

![Figure A2 – IE8 Add-Ons Management Interface](image)

DEP/NX View & Crash:

From Windows Vista and above, Microsoft embedded new feature to Process tab of Windows task manager to view which running process has DEP feature enabled. This feature on Window Task Manager is not available on Windows XP and below.
SysInternal from Microsoft released software called Process Explorer (Figure A3) that has this feature and it is available for Windows XP [4].

Consider a plug-in in Internet Explorer 8 (plug-ins are written using ActiveX Controls) that is vulnerable and installed in victim’s browser. When the attacker tries to execute malicious code on the victim’s browser in the vulnerable plug-in process/memory space, he causes Internet Explorer to crash and force the victim to restart the browser. After the restart of the browser, Internet Explorer 8 crash recovery system won’t recover all the tabs instead it shows the error message below (Figure A4) in those tabs that crashed due to that vulnerable plug-in.

Figure A3 – Process Explorer by SysInternals

Figure A4 – IE8 Error Message After Crash
This security mechanism prevents the malicious code from having multiple chances to exploit the vulnerable plug-in.

**CPU Prevents Malicious Code Execution:**

New CPUs offer new features that prevent a class of exploit where targeting non-executable memory pages to execute arbitrary code. There is a bit that needs to set to enable this feature. This bit is called Execute Disable, XD, in the Intel CPU Family and called No Executed, NX in AMD Processor. Any attempt to execute code in a non-executable area of memory or memory page that has PAGE_READWRITE flag not set causes the CPU to throw an exception STATUS_ACCESS_VIOLATION (0xC0000005).

By default hardware-enforced DEP marks all the memory pages as a non-executable page unless the location explicitly holds executable code. This feature can be configured in system-wide (all processes) or per process. There are four different ways to enable this feature [4, 5]:

- **Opt-In:** This method enables DEP hardware-forced for all the processes that are opt-in to DEP.
- **Opt-Out:** This method enables DEP hardware-enforced for all the processes except of the ones that are opt-out to DEP.
- **Always On:** This enables the DEP hardware-enforced for all the processes regardless of the process supports DEP.
- **Always Off:** This disables the DEP hardware-enforced for all the processes.

**Address Space Layout Randomization (ASLR):**

Microsoft, starts with Windows Vista introduced a new feature called Address Space Layout Randomization or short for ASLR and by default this feature is enabled. The sole purpose of this new feature is to protect the Operating System (OS) from Buffer Overflow. This new feature makes it harder for an attacker to create an automated attack
such as “return-to-libc” where attacker’s code attempts to call system functions during a boot time, because this randomization gives an attacker a probability of $1/256$ to guess the address [2, 3].

Module’s Address Reported by OS (Before Reboot):

<table>
<thead>
<tr>
<th>Module</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel32.dll</td>
<td>0x77AF0000</td>
</tr>
<tr>
<td>user32.dll</td>
<td>0x771E0000</td>
</tr>
<tr>
<td>gdi32.dll</td>
<td>0x75DF0000</td>
</tr>
</tbody>
</table>

*Figure A5 – ASLR Before Reboot*

Module’s Address Reported by OS (After Reboot):

<table>
<thead>
<tr>
<th>Module</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel32.dll</td>
<td>0x77350000</td>
</tr>
<tr>
<td>user32.dll</td>
<td>0x779B0000</td>
</tr>
<tr>
<td>gdi32.dll</td>
<td>0x77A50000</td>
</tr>
</tbody>
</table>

*Figure A6 – ASLR After Reboot*

Figure A5 and A6 shows that every time a machine is rebooted, the OS loads its core modules into a different memory location.

**Extended Validation Certificates:**

In February 2007, Microsoft’s Internet Explorer, like other browsers such as Chrome and Firefox, shaped up their Secure Socket Layer (SSL) to be more visible to the user not just visually by moving it into the address bar but also providing a great deal of information about webpage SSL certificate trust [11, 20].

Starting with Internet Explorer 7, Microsoft used this technology to improve their Phishing Filter by notifying users about the validity of the SSL certification. Users can now trust the identity of the website [38].