CHAPTER 9

Additional Techniques for Aligning Security and Usability

The previous four chapters looked in detail at a variety of design techniques for aligning security and usability. This chapter briefly considers some other approaches that appear promising.

9.1 Additional Patterns for Enhancing Secure Operations

In addition to the patterns that have been previously discussed, these five additional patterns were identified during the research involved in this dissertation:

- **WARN WHEN UNSAFE** (page 345)
  Occasionally it is necessary for users to enter unsafe configurations so that they can accomplish extraordinary operations. The system should periodically warn the user if the system is in an unsafe configuration or engaged in unsafe actions, because users frequently forget to restore the safer configuration.
- **DISTINGUISH SECURITY LEVELS** (page 346)
  Because computer systems typically have multiple security levels at which they can operate, it is important to distinguish those levels to users. Such distinguishing needs to be done in a manner that is consistent both between and within applications (an application of the CONSISTENT CONTROLS AND PLACEMENT and CONSISTENT MEANINGFUL VOCABULARY principles.)
- **DISTINGUISH BETWEEN RUN AND OPEN** (page 343)
  Today's computers use the same interaction gestures to open a document and run a program. Yee discusses that this pun has been responsible for the propagation of worms and viruses in the past.[Yee05a] An approach for minimizing this problem is to distinguish the two commands so that they are no longer identical. Other approaches are discussed in Section 9.3.2.

- **INSTALL BEFORE EXECUTE** (page 342)
  Another technique that can be used to mitigate the threat of hostile programs is to require that all programs be installed before they can be executed. Several approaches for performing this are discussed in Section 9.3.1.

- **DISABLE BY DEFAULT** (page 344)
  Today's computers have an incredible amount of functionality that is never discovered nor needed by most users. Because it is not cost effective to test thousands of different configurations, the functionality needs to be provided so that the few users who need it will be able to use it. However, this functionality can be disabled by default, and only enabled when it is needed.

The remainder of this chapter will discuss the support for these patterns.

9.2 Other Applications of User Auditing

Chapters 3 and 4 of this thesis establish the importance of user auditing for preventing the accidental release of confidential information. According to that principle, information contained within a computer system should be directly auditable by the user: just as the Fair Information Practices [UDoHoAPDS73] prohibit secret record-keeping systems, there should be no secret data contained in our personal computers.

The USER AUDIT pattern can also be applied to data collection and to security states, as shown below.

9.2.1 Auditing physical objects: Apple iSight

An example of *User Audit* applied to physical data collection in computer peripherals is the shutter of the Apple iSight video camera.

Many low-cost USB and Firewire cameras sold to the consumer market do not have a physical shutter, but are instead turned on and off through software running on the host computer. The problem with this design is that it isn't possible to know if the video camera and its built-in microphone are actually recording or not. Indeed, the W32/Rbot-GR computer virus[Sop04, Ley04a] is a computer worm that specifically turns on the victim's web cam and microphones after it has been installed and sets up a video server—quite the thing for the prurient computer hacker.

Apple's iSight Firewire camera has a user-controllable physical shutter. Turning the front of the iSight's housing causes both the shutter to close and the switch to open. Whereas the inside of the iSight's lens is normally a dark grey or black color, the shutter is bright white, making it easy to see—even from across the room. The bright color makes it easier for the user to verify that the shutter has in fact closed.

Apple advertises this pro-privacy feature on the company's web site:

Figure 9-1: Apple's iSight camera is designed to sit atop a computer screen and point directly at the user. The camera includes a built-in microphone.

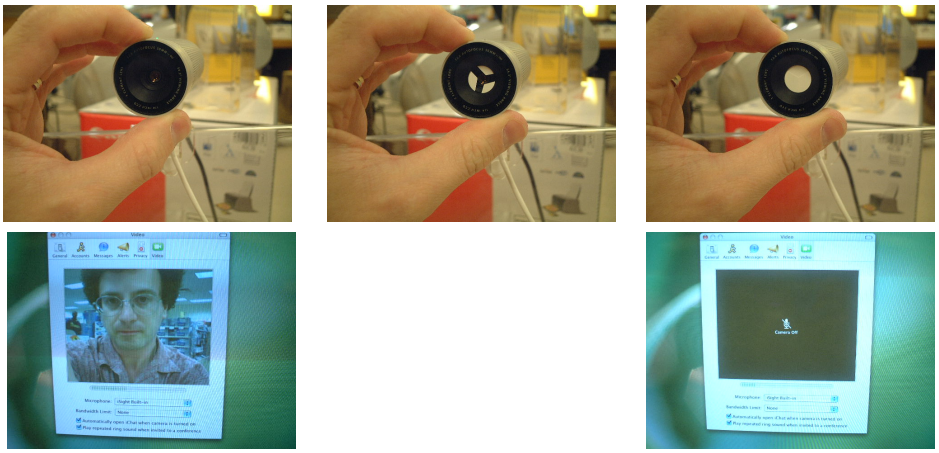

Figure 9-2: The Apple iSight video camera has a shutter that can be opened or closed by turning the front of the camera's aluminum housing. Closing the shutter turns off the camera's video and audio feeds.

> **"Video muting, too**
> Need a moment offscreen to touch up your hairdo or prepare a surprise? Closing the lens cover mutes the video but doesn't disconnect you from your conversation. To resume visual contact, just reopen the lens."[Com04c]

Other devices can support user auditing when microphones are enabled, but not all do. For example, the AT&T ISDN 7506 telephone instruments that are common at MIT, have a speaker phone capability; when the speaker phone is engaged, the little green light next to the "Speaker" button is lit. As a result, it is possible to look at the phone and visually determine if someone is using the phone to listen to the room or not.

On the other hand, the Apple Macintosh PowerBook G4 has a small, high-quality microphone built into the keyboard, but there is no small light or other indication to tell the user if the microphone is listening or not. In fact, the microphone can be turned on programmatically, without the user's knowledge or any visible indication on the computer's screen. Like most notebook computers, the PowerBook is really a marvelous tool for wiretapping a room.

Tang discusses the decision of a major workstation vendor (most likely his employer, Sun Mi-

crosystems), to remove a hardware switch from the company's multimedia microphone in the early 1990s.[Tan97]. The workstation's original microphones contained a small battery and a switch to turn the battery (and the microphone) on or off. Users were forever leaving their microphones turned on and running down the battery, so a revised design had the microphones powered from the workstation itself. At the same time the hardware group decided to remove the switch, for a cost savings of 25 cents per microphone. Tang and other designers were furious, as now users of the workstation would have no obvious way to know if they were being audibly monitored or not. He launched a series of online discussions within the company, but ultimately failed to have the decision reversed. Several years later, however, when the company introduced a video camera for its workstation, the camera came with a physical shutter—in the interest of giving workstation users a physical means by which they could determine whether the camera was watching or not.

9.2.2 User auditing on local systems can promote remote user auditing

In the case of web browser cookies, user auditing on the local system may demonstrate the need for user auditing on remote systems. For example, web cookies can contain information that is difficult or impossible for the browser user to decipher, as shown in Figure 9-3. In fact, the `2o7.net` domain is registered to Omniture, Inc., makers of the SuperStats "Web Site Intelligence" web reporting system. No privacy policy on the Omniture web site indicates what is done with these cookies. A call to the company's headquarters on April 13, 2005, revealed that Omniture does not have a chief privacy officer or any person responsible for privacy issues.

9.2.3 Visually distinguish more-secure from less-secure operations: the SSL lock

The graphic of a lock that is displayed in a web browser is also an example of user auditing—but a troubling example. It is an example that shows how difficult user auditing is to do properly.

In an effort to promote the proprietary encryption technology built into its web browsers and servers, Netscape Communications designed its web browsers to display the icon of a key in the browser's status bar when pages were delivered to the browser using the SSL encryption protocol. A key with one tooth indicated that the page had been delivered with a cipher that used a 40-bit key, while a key with two teeth indicated that the page had been delivered with a cipher that used a 128-bit key. Microsoft copied the idea of using an icon to encrypt security in its Internet Explorer 3.0 browser, but used an icon of a lock instead of a key. As Microsoft's browser achieved market dominance, Netscape was forced to adopt Microsoft's symbology to decrease customer confusion.

The Open Source Mozilla browser also uses the icon of a lock, but adds the icon of an open lock for pages that are not encrypted. Mozilla Firefox shows a blank region instead of an open lock, but keeps the closed lock, displaying it in both the status bar and in the browser's address bar.

Many problems with the lock icon have been identified:

- The lock doesn't really address the secure transmittal of potentially confidential information. The lock tells users that the contents of the web page was delivered securely. But the lock doesn't tell user if clicking a button on the web page will cause the contents of a form on that page to be *sent with encryption*. Early browsers didn't warn if a form that was delivered securely would send back its information without encryption—that is, if a form delivered via an `https:` URL had an HTTP `FORM ACTION` with an embedded `http:` URL.

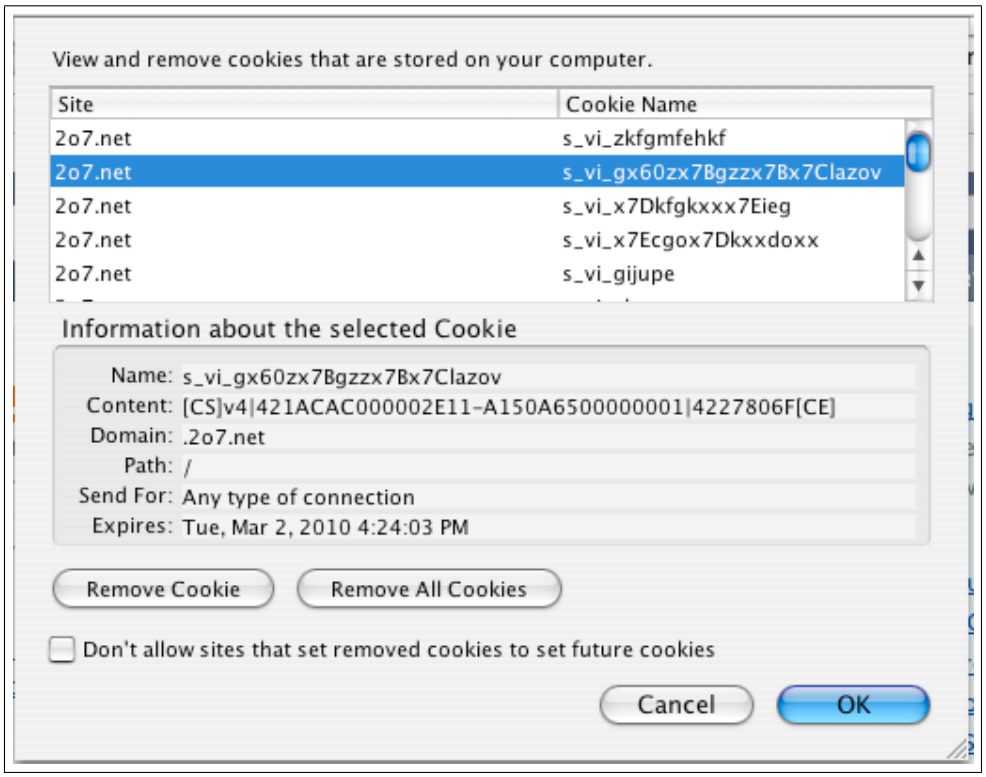

Figure 9-3: The Firefox web browser allows the user to audit cookies stored on the local computer and selectively remove them. Unfortunately, some cookies encode information in a way that is not readily discernible, as is the case with these cookies from the web site `2o7.net`. Ideally these cookies would somehow point to the web site's privacy policy that governs their use. In practice, the web site `207.net` did not even have a privacy policy.

- The lock is not properly integrated with browser prompts for username/password combinations that result from Basic Authentication. By design these prompts are displayed on modal pop-up windows (or, in the case of Safari, on pull-down "sheets"), but these windows do not have provisions for displaying the lock icon. As a result, users of browsers such as Internet Explorer and Mozilla have no easy way of knowing if a typed username/password combination will be sent with or without encryption. This is especially relevant because many web sites implement multiples layers of redirection when switching part of the web site that does not require authentication to a part of the web site that does require authentication.

  The only web browser that appears to indicate whether or not a username/password will be sent with encryption is Apple's Safari web browser. Safari warns the user "your password will be sent in the clear" if the Basic Authentication challenge will be sent without encryption. Unfortunately, it is very doubtful that most users understand what this warning means. (MIT Information Services asserts that the message from Safari is, in fact, erroneous when Safari is used with MIT's TechTime personal Calendar.[MIT03])

  Usability could be improved through the showing of the SSL lock on the Safari username/password panel.

- Although users were instructed to look for a lock before trusting their credit cards to a web

```
% whois 2o7.net
...

Registrant:
Omniture, Inc.
   550 East Timpanogos Cir
   Building G
   Orem, UT 84097
   US

   Domain Name: 2O7.NET

   Administrative Contact:
      MyComputer.com            dnsadmin@MYCOMPUTER.COM
      1358 W BUSINESS PARK DR
      OREM, UT 84058-2203
      US
      801-722-7000 fax: 801-722-7001

   Technical Contact:
      Network Solutions, LLC.           customerservice@networksolutions.com
      13200 Woodland Park Drive
      Herndon, VA 20171-3025
      US
      1-888-642-9675 fax: 571-434-4620

   Record expires on 29-Sep-2005.
   Record created on 29-Sep-2000.
   Database last updated on 13-Apr-2005 14:37:57 EDT.

   Domain servers in listed order:

   NS1.OMNITURE.COM             216.52.17.51
   NS2.OMNITURE.COM             216.52.17.52

%
```

Figure 9-4: The 2o7.net web site is registered to Omniture, makers of the SuperStats Web Site Intelligence product.

site, they weren't told *where to look*. In recent years a growing number of web sites have been copying the SSL lock and placing it in the body of their web pages. Gutmann suggests that the lock is a kind of talisman that the web designers now display in the graphics of the page itself to engender customer confidence. [Gut04b]

- The protection offered by SSL can be circumvented by browser plug-ins or (in the case of Internet Explorer) browser helper objects, which have access to form data before the information is encrypted by the SSL layer. Thus, SSL does not protect against spyware
- Finally, whether or not a page of information will be sent or received with encryption has no bearing on the security of the web server on which the information resides. In more than 10 years of online commerce there is not a single recorded instance of a credit card being captured while being transmitted from a web browser to a web server. On the other hand,

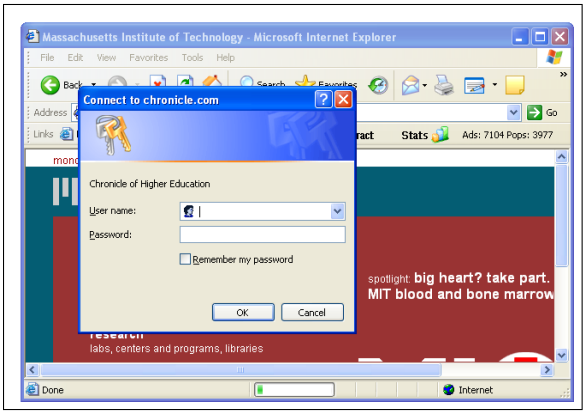

Figure 9-5: This figure shows the transition from a page that is not using SSL encryption to one that is using SSL encryption and for which the password is going to be sent using the SSL encryption protocol. This image was created on Microsoft Windows with Internet Explorer 6

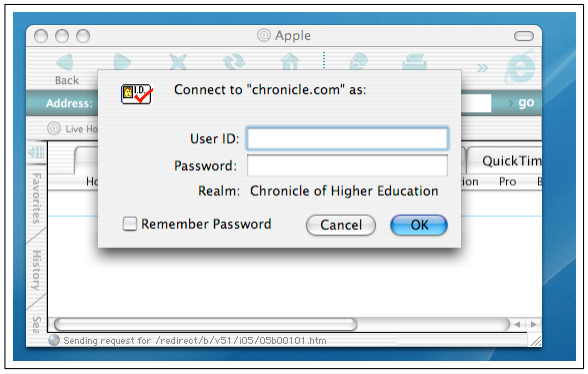

Figure 9-6: The transition from an unencrypted page to an SSL-encryption page that requires authorization using Internet Explorer 5.2 on the Macintosh

Figure 9-7: The third screen shot of the transition series. This screen shot created on a Macintosh using Firefox 0.9.1

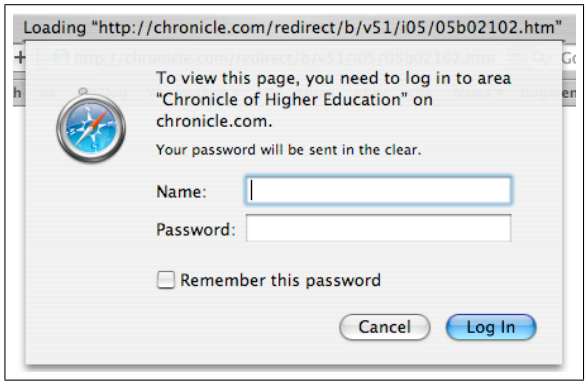

Figure 9-8: The final screen shot of the series. This one is created on a Macintosh using Apple Safari 1.1. Although the panel indicates that the password will be sent "in the clear," it is unlikely that many of Apple's users understand precisely what "in the clear" actually means.

> SSL offers "no protection for cards once they're at the merchant's server." Theft of credit card numbers from unsecured servers is rampant; as a result, the black-market price for a million stolen credit cards has dropped from $100 a few years ago to less than $1 today.[Gut04b]

These problems loosely group into two categories. First, the lock icon can give a false sense of security. Second, the visual indication isn't present where it is needed.

Because browsers have two fundamentally different ways of sending and receiving data over the Internet—one with encryption, one without— it is good that browsers can give users a visual indication of the two states. Alas, one of the persistent problems with the SSL "lock" icon is that

it shows if a page was *received* with SSL; the lock does not indicate whether or not forms that are submitted will be *sent* using SSL. In some cases browsers give a warning when there is a transition between the SSL and non-SSL states; in other cases they do not.

9.3 Operating System Improvements

This section explores two operating system improvements that would likely increase secure operations with minimal impact on security.

9.3.1 Install before execute

Kirovski, Drinic and Potkonjak[KDP02] observed in 2002 that users rarely if ever need to run programs that have not been properly installed. Reid made a similar observation in 1987 when he wrote "Nobody, no matter how important, should have write permission into any directory on the system search path. Ever. One should not be able to install a new program without typing a password."[Rei87] Garfinkel and Spafford recommend against placing the current directory (".") in the Unix search path, insisting that the only programs that should be runable without typing a full pathname are those that have been properly installed.[GS91, p.152]

The system proposed by Kirovski *et al.* uses a **Trusted** mode in which software cannot be run unless the software has been properly installed. In the scheme that the trio present, every processor is equipped with an unchangeable and highly protected unique identifier. This identifier is used to perform a series of transformations on each block of instructions when the application is installed. These transformations are similar to using steganographic techniques to store information in an instruction stream. When a program is run, the processor verifies each block as it is loaded into the instruction cache to verify that the stored number matches the processor's unique identifier. Thus, this technique is robust against techniques such as stack and heap attacks that can inject untrusted instructions into the memory space of formerly trustworthy programs: once instructions are injected, the instruction blocks will no longer verify and thus no longer run.

Although not as powerful as the approach that Kirovski *et al.* present, there are other ways to implement the INSTALL BEFORE EXECUTE pattern on conventional desktop operating systems and microprocessors:

- The operating system could refuse to run programs that are not located in explicitly specified directories. For example, most Windows applications are installed inside the `\Windows` or `\Program Files` directories. The operating system could refuse to run executables unless they were contained within these directories, and prohibit writing into these directories except as part of an explicit installation process.
- Protection could be accomplished with the use of execute permissions similar to the Unix file permission. Such a system would only allow these permissions to be set as part of a trusted installation process.
- The system could refuse to run programs that were not digitally signed with a host-specific key. Code would be signed as part of the installation process.

One problem with these approaches is that they do not protect the computer against executables

that include interpreters. For example, a properly installed and verified copy of Microsoft Word would nevertheless be able to run a Word Macro Virus. Nevertheless, eliminating the ability to run native code that had not been properly installed would be a big step forward on desktop operating systems. Such mechanisms could also help further the goals of the "Pure Software" proposal made in Section 8.3.

9.3.2 Distinguish between running programs and opening files

Yee observed that typical icon-based interfaces do not distinguish between *viewing* and *executing*: both actions are initiated with a double-click. Double-clicking on an executable runs that executable; double-clicking on a non-executable instructs the operating system to determine the appropriate application for that executable, run it (if it is not running), and send the non-executable's file name to the executable in a message.[Yee02, Yee04] This is another example of Neumann's dangerous computer "puns." [Neu90]

It is an open question as to whether or not *users'* mental models distinguish between viewing and executing. It is likely that frequent users of programs like Microsoft Word distinguish between executing the Word application and viewing or editing a document that they are working on. On the other hand, users may not distinguish between running a web browser and viewing a remote web site; the distinction may even break down in the case of a web site that contains Java applets or other active content.

Yee suggests that a consequence of this confusion between running and opening is that users occasionally run programs without intending to do so. For example, the "Love Letter" worm released in May 2000 was named `LOVE-LETTER-FOR-YOU.TXT.VBS`.[CER00] Microsoft Windows frequently displays files without showing their extension, causing this hypothetical program to display as `LOVE-LETTER-FOR-YOU.TXT`. Thus, many users thought that "opening" the file `LOVE-LETTER-FOR-YOU.TXT.VBS` would actually run the Windows Notepad and show the contents of the file `LOVE-LETTER-FOR-YOU.TXT`; instead, the program `LOVE-LETTER-FOR-YOU.TXT.VBS` launched, which then proceeded to do its mischief.

9.4 Eliminating the Security Policy "Construction Kit"

A current trend in security interfaces is to give users the ability to fine-tune security policy for their own particular needs. This trend has resulted in interfaces that exhibit the problem of hyperconfigurability: the interfaces have dozens of controls for manipulating minutiae of policy enforcement. The security interfaces become, in essence, security policy "construction kits," rather than tools for selecting between a small number of well-thought-out and tested policies.

For example, the Microsoft Internet Explorer 6 Service Pack 2 "Security Options" panel has 38 different controls for manipulating security policy. Some of these controls have binary choices, such as "Open files based on content, not file extension" which can be set to "Disable" or "Enable." Others have three values, such as "Submit nonencrypted form data," which can be set to "Disable," "Enable" or "Prompt." Given the large number of controls, the single dialogue presented in Figure 9-9 represents a configuration space of $2^{10} \times 3^{21} \times 4^2 \times 5^1 = 856,912,134,389,760$, or roughly 2^{50} possible configurations.

The 2^{50} number may be a significant underestimate, however, as many of the choices in the Figure 9-9 dialogue actually refer to additional constellations of security policy choices configured on other panels.

Hyperconfigurability is not in the interest of Microsoft nor its customers. It complicates developing, testing, documentation, validation, and maintenance. Furthermore, because many of the choices that are presented to the user are not orthogonal, it is likely that choices could be simplified or removed without a loss of expressiveness.

Many of the configuration options on this panel could be collapsed. For example, it would seem that there is little reason to distinguish running unsigned .NET Framework-related components from running unsigned ActiveX controls.[1] A single control could allow or disallow the running of both unsigned controls and components.

Hyperconfigurabilty is present in many different security systems. For example, Farmer has suggested that "system security degrades in direct proportion to use."[Ros05] Farmer says that this is true with firewalls, for example, because users of firewalls steadily change policies by opening holes whenever they need to get a new application or protocol to work, but they never go back and close the hole when it is no longer needed. Users also find it easier to make policy changes that lower security, rather than to find work-arounds to accomplish their goals within the established security framework.[Far96] Such policy adulteration is made possible by hyperconfigurability.

9.4.1 Explaining hyperconfigurability

A series of interviews were conducted at Microsoft's Redmond Campus in January 2004 to understand why the Windows operating system includes such pervasive support for hyperconfigurability. Discussions about hyperconfigurability were held with several Microsoft employees who were at the time directly involved in security and application design. Among these employees was the manager of Microsoft's Windows XP Service Pack 2 project. Some of the explanations for hyperconfigurability include:

- **Defensive Product Development.** Because Microsoft's security practices are frequently criticized by those outside Microsoft, when the company introduces a new technology into Windows or Internet Explorer, it needs to create an accessible control that can be used to explicitly turn the technology off.

  Simply allowing the technology to be disabled through the use of a registry setting is generally not sufficient: doing so would leave Microsoft vulnerable to criticism. Even a master control that would turn off all new and potentially dangerous technologies would leave Microsoft open to criticism by pundits who didn't realize that the master "off" control also turned off the new technology.

- **Vestigial Controls.** As Microsoft develops new technology, it is faced with the question of

[1] Indeed, the wisdom of ever allowing unsigned ActiveX or .NET components should be questioned. Although it is tempting to think that developers require the ability to run unsigned components while the components are under development, it would be easy for the Microsoft development environment to create a self-signed certificate upon installation and then for the linker to automatically sign components as part of the linking process. Even without third-party attestation of identity, such certificates would allow users to determine if a new component came from a developer who had provided a previous version of the same component.

what to do with its old security controls. Although it might be more elegant to update its old controls with new functionality, doing so may break backwards-compatibility with existing applications (including in-house applications written by major customers), documentation, training materials, and third-party web sites. To take the example introduced above, if Microsoft had relabeled the radio-buttons that control the running of unsigned ActiveX controls with a new radio-button that handled all unsigned components, existing web sites with instructions on how to configure Internet Explorer for "higher security" would suddenly be incorrect. Customers reading these web sites might become confused.

Microsoft's employees owe much of their company's success over the past 20 years to the company's long-term insistence on maintaining backwards compatibility. Many DOS and Windows 3.1 programs will still run on Windows XP, for example, as XP has support for many DOS interrupt vectors and the Windows 3.1 16-bit APIs. (For example, a copy of the DOS game `rogue.exe` from 1983 will run in the `cmd.exe` command box of a Windows XP Professional system in March 2004.) It is not at all surprising that this concern for providing backwards compatibility would extend to preserving human interfaces wherever possible.

- **Customer Security Needs.** Many of Microsoft's enterprise customers have security groups that want to manage specific aspects of the security policy of their desktop and server operating systems. Some of the fine-grained security policy controls in Windows are the result of customers seeing beta releases of new Microsoft operating systems and telling Microsoft that the security group would not allow that new version of Windows to be deployed unless the ability to disable a new feature was specifically added.

  Based on interviews, it was not clear if Microsoft had a means for formally tracking which customers required which features, or for contacting those customers at a later point in time to see if the control could later be removed. Indeed, the *Vestigial Controls Problem* discussed above implies that such controls, once they are incorporated into the operating system, *cannot ever be removed.*

Hyperconfigurability present in the Windows operating system is not the result of a specific Microsoft policy, but is instead an emergent phenomena based on Microsoft's market position and its development practices.

Hyperconfigurability is not just a problem for Microsoft: it is endemic to the computer industry. Just as Windows has become more complicated with each successive release, so too has grown the complexity of the Macintosh OS X operating system. In fact, the Macintosh has become so complicated that MacOS 10.4 has a search facility to help users to find settings and controls within the computer's extensive set of control panels.

Perhaps the most flagrant example of hyperconfigurabilty is Security-Enhanced Linux from the US National Security Agency.[Nat05] A security policy for an SELinux system consists of more than 10,000 lines of code spread out among 200 different definition files. This policy provides extraordinarily fine-grained control over what the Linux kernel may and may not do. NSA delivered this system without a tool to modify the definition files. Instead, it was expected that system administrators would modify them manually. (Hitachi Software Engineering released a beta version of its SELinux Policy Editor in 2003, but has not updated the system since. The current version of the tool does not support the Linux 2.6 kernel.[Co.03]

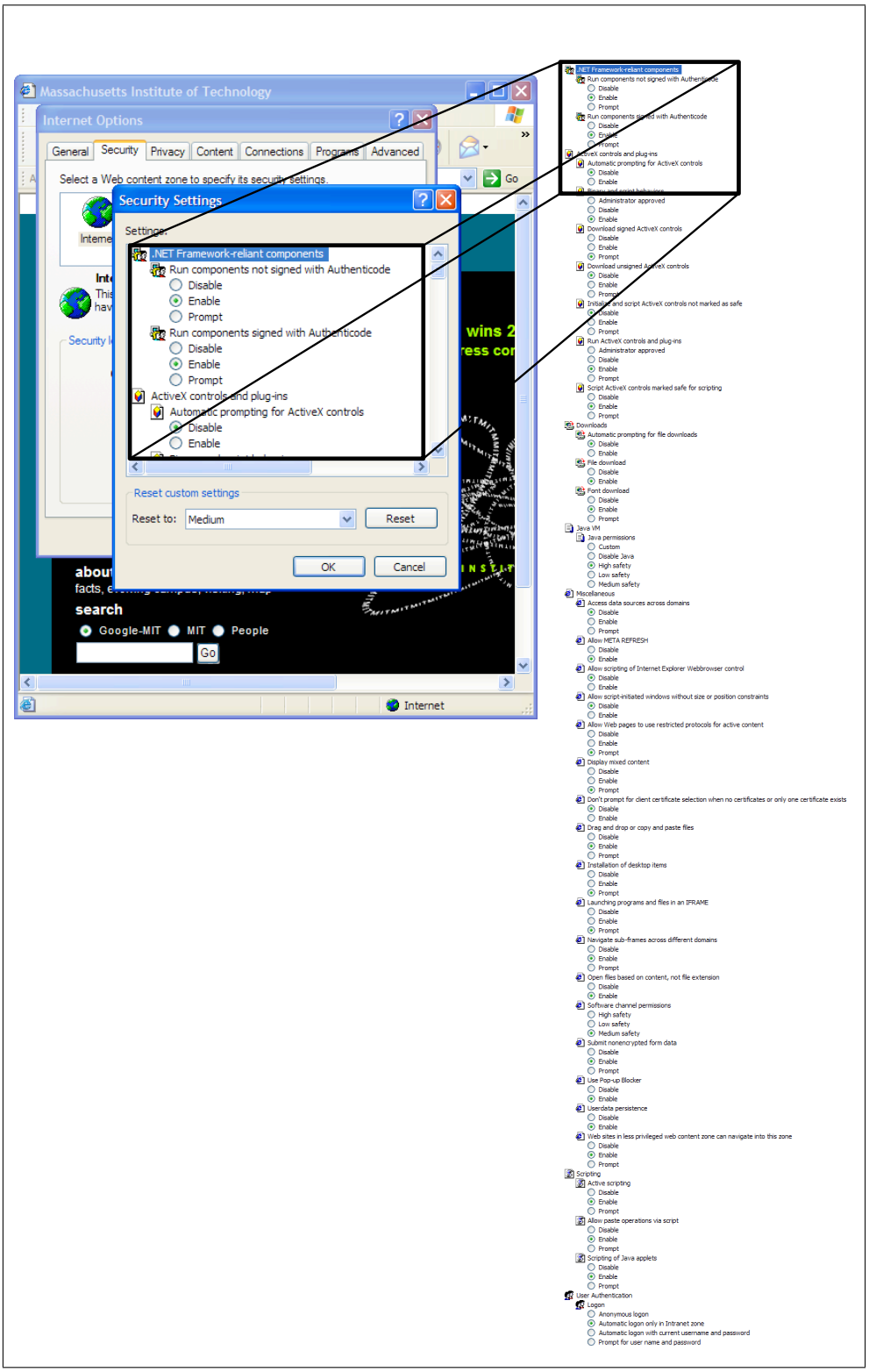

Figure 9-9: Internet Explorer 6 SP2 has a wide variety of security options. The total configuration space is approximately 2^{50} distinct states, although some of these states are degenerate.

Hyperconfigurability increases the appearance of security, but does not increase actual security. Instead, it is more likely that security will be increased by having a small number of well-understood configurations. If a specific organization refuse to adopt new technology without the ability to tweak security configurations, it may be appropriate to provide those organizations with their own security tweaking tools, such as the Microsoft "PowerToys for Windows XP" collection.[Cor05d]